Patrocinado por

COR Technologies
Consultora en Capacitación Informática
Consultora en Seguridad Informática

WWW.CORTECH.COM.AR

distribución gratuita



n°8

MAYO 2004

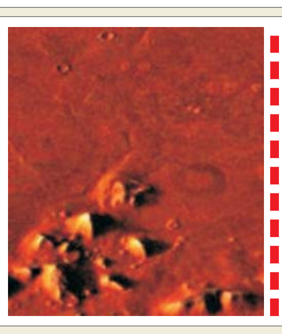
IBM Mainframe

Desde el nacimiento de las PCs, existieron rumores que los Mainframes de IBM iban a dejar de existir. Hoy, 40 años después, siguen en vigencia. En este artículo veremos su historia (nacimiento y evolución), presente y futuro de este hardware que no está al alcance de todos.



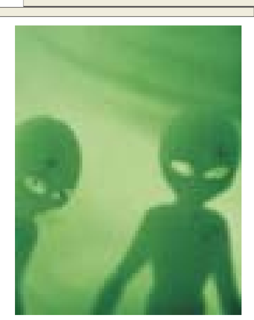
Modelos de Negocio Open Source

En este artículo se ofrece una visión del software libre desde el punto de vista del emprendedor. Se van a recorrer algunos modelos de negocio relacionados con el software libre con viabilidad comprobada en el mundo real.



PKI

Se exponen los conceptos de criptografía y Public Key Infrastructure (PKI) que son la piedra fundamental para el entendimiento de muchísimos conceptos relacionados a la seguridad en IT.



SCRIPTING

Perl

VBScript

WSH

ADSI

WMI

editorial

NEX8 presenta un abanico de temas que creemos serán de su interés. Hemos recibido varios e-mails solicitándonos abordar el Scripting. Y eso hemos hecho en dos de nuestros artículos de NEX. Lo hemos dividido en dos escenarios complementarios: scripting en el mundo tipo-UNIX (con un claro ejemplo Perl) y Scripting en los sistemas operativos de Microsoft (a partir de Windows 2000). Esperamos que quien lea esos artículos saque una idea clara de la utilidad de los scripts sumado a entender la filosofía general. Además, como siempre damos referencias bibliograficas muy seleccionadas para que quien desee indagar más le sea un poco mas fácil. La idea fue mas bien ver el scripting del lado de “administrador de sistemas”. Hemos dejado para NEX9 lo que serian tecnologías de scripting para el mundo de desarrolladores web. Por primera vez hemos incluidos dos artículos que no son tan técnicos como es nuestro perfil. Uno apunta a la historia de las computadoras: los 40 años de la primera mainframe de IBM. Mucho hemos visto en los medios del exterior sobre este acontecimiento que es orgullo de IBM pero muy poco en medios locales. De alli el homenaje a la S/360. El segundo artículo no techie tiene que ver con entender el modelo de negocios que plantea el software no propietario. Este es un excelente articulo escrito por Fernando Monera Daroqui (presidente de Opensistemas, (www.opensistemas.com) basado en su presentación en el ultimo congreso HispaLinux (una de las reuniones más prestigiosas del mundo Open source de habla hispana), en Septiembre de 2003,.
El suplemento de seguridad continúa complementando nuestro conocimiento de seguridad informática. En NEXes anteriores habíamos visto “elementos de criptografia” y “kerberos”. El paso natural es entender la infraestructura de manejo de certificados que nos propone PKI que desarrollamos en detalle junto al caso particular de PGP (Pretty Good Privacy). No se pierdan de conocer la vida de R. Zimmermann , su inventor.
Solo me queda mencionarles dos artículos: un análisis de los mejores mail-servers bajo Linux y conocer el Small Buiseness Server (SBS) 2003 de Microsoft.



Staff

Año 3 - Número 7 - Abril 2004

Director
Dr. Osvaldo Rodríguez

Propietarios
COR Technologies S.R.L.

Coordinador Editorial
Carlos Rodríguez Bontempi

Cordinación General
María Luján Zito

Responsable de Contenidos
Dr. Osvaldo Rodríguez

Editor en Jefe
Raúl Kuzner

Redactores
Martín Sturm, Javier Pierini, Raúl Kuzner, Osvaldo Rodríguez, María Luján Zito, Leonel F. Becchio, Rodrigo M. González, Hugo Cela, Guido Lorenzutti.

Humor
Marcos Severi

Distribución
Paola Karvouniaris, Ximena Antona

Diseño Web Site
Emanuel A. Rincón

Diseño Gráfico
Víctor Pereyra
Carlos Rodríguez Bontempi

Publicidad
Ximena Antona
publicidad@nexweb.com.ar
+54 (11) 4312-7694

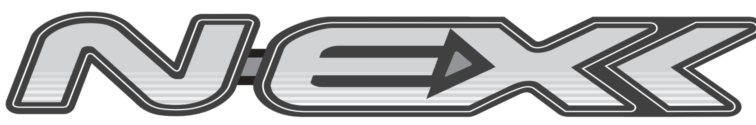
Preimpresión e Impresión
Talleres Gráficos S.A.
Buenos Aires Herald Ltd.Ind.Fin.
Azopardo 455
C1107ADE - Capital Federal

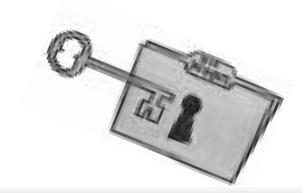
NEX - Periódico de Networking
Registro de la propiedad Intelectual en trámite leg3038

Dirección: Av. Córdoba 657, Piso 12
C1054AAF - Capital Federal
Tel: +54 (11) 4312-7694
http://www.nexweb.com.ar

El staff de NEX colabora ad-honorem, si desea escribir para nosotros, enviar un e-mail a:
articulos@nexweb.com.ar

Retire su ejemplar NEX en forma gratuita en Av.Córdoba 657, Piso 12 - Capital Federal o solicítelo telefónicamente para su empresa al +54 (11) 4312-7694 http:// www.nexweb.com.ar





Modelos de negocio basados en software libre

Es muy importante lograr entender las relaciones entre el software libre y la empresa. En este artículo se pretende ofrecer una visión del software libre desde el punto de vista del emprendedor. Se van a recorrer algunos modelos de negocio relacionados con el software libre con viabilidad comprobada en el mundo real. Sabemos que existen multitud de modelos y variantes que no vamos a tratar ya que posiblemente excedería el objetivo del artículo.

1- Introducción

El software libre ha experimentado un crecimiento muy importante en la última década. Al principio apoyado por personas que vendían ideas, como R.M.Stallman o E.S.Raymond, construyendo una percepción del software libre alrededor del concepto de «Hacker» como programador excepcional motivado por la calidad de su código. Se hablaba entonces de las posibilidades de Linux y otros proyectos de software libre de perdurar en el tiempo y de convertirse en lternativas reales de propósito general. Era necesario alcanzar una determinada masa crítica de usuarios que garantizara la supervivencia del modelo y su crecimiento.
Casi desde el principio aparecieron intentos de poner en marcha negocios alrededor del software libre aunque por regla general no tuvieron excesivo éxito (honrosa excepción de algunas empresas como RedHat). Se cuestionaba bastante la viabilidad de negocios alrededor del software libre, tras fracasos importantes de empresas como VALinux o Corel.
En los últimos tres o cuatro años el concepto de Software Libre asociado a la imagen de «Hacker» ha cambiado radicalmente. Se ha producido una revolución muy importante que consiste en la adopción de grandes proyectos de Software Libre (en especial Linux, Apache y MySQL) por parte del entorno empresarial. Esto ha supuesto para el Software Libre la capacidad de ofrecer parte de las necesidades que el concepto «Hacker» no podía proporcionar y que el entorno empresarial demandaba, como soporte profesional, acuerdos de nivel de servicio, compatibilidad con otras plataformas o certificaciones hardware y software.
Hoy en día ya no se habla de la masa crítica de usuarios, ni de la viabilidad empresarial del software libre. Este nivel se ha alcanzado ya. En este momento existen multitud de modelos de negocio alrededor del software libre con viabilidad demostrada y probada. En este artículo se pretende ofrecer una visión del software libre desde el punto de vista del emprendedor. Se van a recorrer algunos modelos de negocio relacionados con el software libre con viabilidad comprobada en el mundo real. Sabemos que existen multitud de modelos y variantes que no vamos a tratar ya que posiblemente excedería el objetivo del artículo.
A medida que el Software Libre vaya creciendo en número de usuarios y empresas que lo utilizan irán apareciendo necesariamente nuevos modelos de negocio no recogidos en este artículo. El presente artículo sirvió como prólogo a la ponencia que se presentó en el ultimo congreso hispaLinux en Septiembre de 2003, cubriendo el análisis de los diferentes modelos de negocio viables dentro del mundo del Software Libre, considerando como marco de referencia el avance experimentado por estas tecnologías en los últimos años y desde una óptica de comparación con el software distribuido bajo licencia propietaria, ofreciendo un imagen presente y una valoración futura de la viabilidad de estos modelos de negocio.

2 - Cambio de planteamiento: de producto a servicio

Desde el punto de vista empresarial, el software libre supone un cambio de planteamiento con respecto al software propietario tradicional.
El cambio fundamental consiste en el paso de obtención de ingresos por venta de productos a obtención de ingresos por venta de servicios.
En general, el concepto de software libre invalida la obtención de ingresos por repetición de ventas de licencias de uso de un mismo producto cerrado, ya que, por su propia naturaleza, cualquier persona que obtenga ese software es capaz de modificarlo y copiarlo dentro de las restricciones impuestas por cada licencia. Ninguna empresa pagaría por un producto (recalco lo de producto) si es capaz de obtenerlo gratuitamente a no ser que el desembolso económico suponga una serie de servicios adicionales por los cuales si que estaría dispuesta a pagar.
Mientras que el software propietario basa sus ingresos en las licencias de uso de software, el software libre tiene que buscar su rentabilidad en los servicios asociados a ese software y no en el producto en si.
Uno de los casos más brillantes podemos encontrarlo en la empresa MySQL AB, que analizaremos con algo más de profundidad más adelante. Esta empresa ha construido su modelo de negocio alrededor de un software de Base de Datos (MySQL) el cual, en muchos aspectos (escalabilidad, soporte al cliente final, características técnicas, etc), está por detrás de grandes productos comerciales como Oracle o DB2.
Las posibilidades de sobrevivir de MySQL AB compitiendo con estos otros productos es nula en un ámbito puramente comercial. Pero MySQL es software libre, cualquiera puede instalarlo y utilizarlo, y cubre todas las necesidades de la mayoría de las implantaciones. Esto ha hecho que MySQL se encuentre instalado en más de cuatro millones de sistemas en producción (no es posible conocer el número exacto, ya que nadie vende licencias de uso y es posible la descarga y uso del software casi sin limitaciones). Es impensable que MySQL hubiera logrado esos números compitiendo con Oracle o DB2.
MySQL AB renuncia a parte de los ingresos obtenidos a través de licencias a cambio de incrementar la base de usuarios y obtener ingresos entre otras áreas, como los contratos de soporte y consultoría o las peticiones de personalización que algunos de estos usuarios solicitan a la empresa. Las mejoras logradas en el software son incorporadas al propio producto de modo que pasan a estar disponibles para todo el mundo en la siguiente versión.
Debido a este proceso, la base de datos MySQL está inmersa en un desarrollo muy rápido y está alcanzando en funcionalidad a sus hermanos mayores. Los usuarios empiezan a demandar (e implementar!!!) funcionalidades que solamente existían en grandes productos propietarios (escalabilidad, seguridad, etc.).
El uso de software libre todavía tiene una serie de resistencias que, en algunas empresas, son casi insalvables hoy en día. En las pequeñas empresas todavía existe un cierto temor a los cambios y a la falta de compatibilidad con lo que ha existido hasta ahora. Es por ello que la adopción de software libre está siendo asumida primero primero por las grandes empresas y las entidades gubernamentales.

3 - Ventajas competitivas

La utilización de software libre en el entorno empresarial ofrece una serie de ventajas competitivas muy importantes, pero que suponen un cambio de mentalidad sobre un modelo (el modelo clásico de venta de licencias) asumido completamente por el entorno. Algunas de las principales ventajas competitivas que puede ofrecer la utilización de software libre son las siguientes:

Capacidad de modificación del código: Con software libre una empresa tiene capacidad para adaptar la solución a sus necesidades, arreglar fallos operativos o de seguridad, etc.
Independencia del proveedor: La implantación de una solución basada en software libre permite al cliente la elección del mejor proveedor de servicios. La disponibilidad del código fuente y la capacidad para modificarlo permite que una empresa no quede atada a un determinado proveedor.
Seguridad: Se disminuye o incluso se puede llegar a eliminar la existencia de puertas traseras, troyanos, etc, debido a que cualquier empresa puede auditar el código fuente de las aplicaciones que pone en producción. Adicionalmente la propia «comunidad de usuarios» está desarrollando constantemente el mismo trabajo de auditoría de código.
Garantías de permanencia: La utilización sistemática de estándares hacen difícil que una determinada aplicación pueda quedar sin soporte. En el caso de que esto suceda, la disponibilidad del código permite que otro grupo de usuarios u otra empresa pueda tomar el liderazgo en el desarrollo. Las aplicaciones basadas en software libre solamente mueren cuando dejan de utilizarse, generalmente porque aparezcan aplicaciones superiores que actúen como sustitutivas.

Desde el punto de vista del empresario, el desarrollar software libre también tiene una serie de ventajas indudables, como son:
Disponibilidad de una comunidad potencial enorme de programadores y probadores del software
Posibilidad de respuesta rápida ante clientes por problemas en el código, consecuencia de la anterior
Evolución «automática» de las distintas piezas de software que componen una solución
Tendencia a una calidad enorme en el código desarrollado, como consecuencia de la necesidad de trabajar en equipo con personas desconocidas
Tendencia a la utilización de estándares, lo que permite la construcción de soluciones mucho más completas mediante integración de distintas aplicaciones

Es difícil para una empresa que invierte recursos en el desarrollo de software la decisión de liberar el código. El éxito de la iniciativa depende en gran medida de la utilidad real que el software en cuestión tenga para la mayoría de los usuarios.

4 - Algunos modelos de negocio basados en software libre

En esta sección vamos a comentar una serie de modelos de negocio relacionados con software libre. Evidentemente no están cubiertos todos ellos, ya que casi cada día aparecen nuevas ideas o planteamientos, algunos de ellos muy exitosos.

Con software libre una empresa tiene capacidad para adaptar la solución a sus necesidades, arreglar fallos operativos o de seguridad, etc.

Hemos dejado fuera conscientemente algunos modelos de negocio perfectamente válidos y exitosos como los relacionados con formación o consultoría ya que son esencialmente iguales si están basados en software libre o en

▶▷



Solución del crucigrama de la pag. 22



software con licencia propietaria. No todos los modelos de negocio comentados se basan en software libre puro. Algunos de los modelos de negocio relacionados con software libre más exitosos se basan en la mezcla en mayor o menor medida de software libre y software propietario. Hay un concepto clave para comprobar la viabilidad de un negocio. Los modelos de negocio basados en software libre más exitosos se logran cuando los dos actores principales detrás del desarrollo de una determinada tecnología (el propietario del software y la «comunidad») se encuentran motivados para mantener y hacer evolucionar dicha tecnología (vease más adelante el ejemplo de TrollTech con las librerías QT).

4.1 Software libre como plataforma a software comercial

No se puede decir que este sea un modelo de negocio basado en software libre propiamente dicho, pero ciertamente tiene bastante interés para completar el «gran dibujo» que supone la relación entre el mundo del software libre y el mundo del software propietario.

Algunas grandes empresas comerciales tradicionales como podría ser Oracle, o en su momento Corel, han seguido con cautela la adaptación
al software libre. Con piezas de código cerradas que suponen parte de su «core-business», han intentado (con mayor o menor éxito) la integración con el mundo del software libre tratando de mejorar la compatibilidad y permitiendo la ejecución de su software sobre sistemas operativos libres (como Linux).

Este modelo de negocio consiste básicamente en la adaptación de determinadas aplicaciones disponibles bajo licencia propietaria de modo que pueda coexistir con aplicaciones basadas en software libre. Hoy en día existen multitud de empresas siguiendo este modelo de negocio.

Algunos ámbitos de necesidad todavía no están cubiertos por software libre, aunque rápidamente se están llenando todos los huecos. Muchas empresas cubren estas necesidades desarrollando software propietario capaz de ser ejecutado en sistemas libres como Linux. Hay multitud de ejemplos que podemos citar, como:

- SAP/R3
- Siebel
- Macromedia Flash
- HP openview

Todos ellos están cubriendo necesidades existentes en el mundo del software libre y que no se encuentran todavía cubiertas satisfactoriamente por aplicaciones libres.

Este modelo de negocio no es válido si asumimos como tal la capacidad de obtener ingresos sostenidos de un determinado esfuerzo.

Existen históricamente multitud de ejemplos en el ámbito del software libre que lo corroboran, como por ejemplo:

- ApplixWare o Corel WordPerfect Office, son suites ofimáticas propietarias que pretendían cubrir las necesidades ofimáticas en Linux. Hoy en día prácticamente han desaparecido con el crecimiento de aplicaciones libres sustitutivas.
- SSH, protocolo de comunicaciones propietario, desaparecido rápidamente tras la aparición de openSSH.
- IPlanet Enterprise Server, es un servidor web con licencia propietaria prácticamente en extinción por el éxito de Apache.

Hay algunas aplicaciones que pueden encontrarse en un callejón sin salida si no logran encontrar un nicho de usuarios suficientemente importante en el mundo del software libre, como pueden ser los diversos servidores de aplicaciones, por el éxito de alternativas libres como Tomcat, JBoss o Jonas, sistemas de Firewall propietarios como FW1 o servidores de Directorio LDAP como IPlanet Directory Server.

Este modelo de negocio planteado no logra captar las ventajas competitivas derivadas de modelos de negocio puramente basados en software libre por lo que pueden llegar a tener dificultades para obtener rentabilidad a largo plazo.

Si la necesidad existe y es real, a la larga aparecerán proyectos de software libre que cubran dicha necesidad, con todas las ventajas inherentes que proporciona el uso de software libre.

4.2 Modelo de desarrollo con doble licencia

Este es uno de los nuevos modelos de negocio que se ha comprobado que son exitosos. Básicamente se trata de asociar dos licencias a un determinado código, una de ellas es licencia libre y otra es licencia propietaria. Como usuario, puedes elegir cualquiera de las dos licencias para aplicar al uso del software. Se podría asumir que este modelo de negocio está llegando a la madurez debido a que ha sido probado durante bastantes años con éxito por múltiples empresas.

4.2.1 Trolltech y las librerías QT

Por ejemplo, Trolltech dispone de las librerías QT (base de KDE y de otras muchas aplicaciones). Dichas librerías se distribuyen bajo licencia GPL para UNIX siempre que se vaya a desarrollar software libre con ellas. Para el desarrollo de software comercial Trolltech puede entregar las mismas librerías bajo licencia comercial, de modo que una empresa que no desee liberar el código que desarrolle no tenga la obligación de asignar la licencia GPL a sus desarrollos, sino que tenga la alternativa de obtener una licencia de Trolltech que le permita desarrollar sus aplicaciones sin «heredar» la licencia GPL.

4.2.2 OpenOffice

OpenOffice es una suite de oficina similar a Microsoft Office, capaz de alcanzar unos niveles de compatibilidad de formato que no se conocían hasta su aparición. OpenOffice (originariamente StarOffice) nació como una aplicación de código no libre, que se ofrecía gratuitamente por la empresa StarDivision. Esta aplicación se desarrolló sobre la base de tratar de imitar la suite de oficina de Microsoft, explícitamente con el planteamiento de no reinventar la rueda y hacer que los usuarios se sintiesen cómodos con su uso. Recientemente StarDivision fue adquirida por SUN, la cual un año después liberó openOffice con una doble licencia. La primera era una licencia libre (una variante de la licencia GPL) y la segunda era una licencia propietaria, la cual permite a una empresa utilizar el código de openOffice para crear productos propietarios tras pagar la correspondiente licencia a SUN. Cualquier persona o empresa se puede acoger a la licencia que desee de las dos.

SUN hasta la fecha ha mantenido dos versiones del código fuente original StarOffice:

- StarOffice: Es un software derivado de openOffice que se vende a un precio de alrededor de 75 dólares. Básicamente se trata de openOffice con una serie de software adicional y las fuentes mejoradas.
- OpenOffice: Es la base del desarrollo y se ofrece bajo las dos licencias mencionadas anteriormente. Todas las contribuciones a openOffice requieren que el autor licencie su código bajo las dos licencias mencionadas.

4.3 Soporte y productos alternativos

Este modelo es bastante tradicional y consiste en desarrollar una determinada aplicación bajo licencia libre, ofreciendo personalizaciones y/o servicios específicos sobre este software. Como consecuencia de este modelo de negocio generalmente se evoluciona a la disponibilidad de dos versiones del mismo software, una libre y otra propietaria, ofreciendo esta última una funcionalidad superior.

4.3.1 Sendmail

Sendmail es un software servidor de correo electrónico de amplia implantación. Se decía hace unos pocos años que todos los mensajes que se enviaban por Internet pasaban al menos por un servidor sendmail. No creo que esta afirmación sea válida ya.

Sendmail es un software muy complejo que requiere una amplia experiencia para obtener una parametrización del sistema óptima. La empresa desarrolladora de sendmail ha construido alrededor de este software un modelo de negocio orientado a la consultoría. Ofrece a sus clientes desarrollos a medida relacionados con sendmail, parametrizaciones personalizadas y soporte en general.

4.3.2 MySQL

Uno de los principales ejemplos de este modelo de negocio es MySQL. MySQL es la base de datos libre de mayor implantación. Este software se distribuye bajo la licencia GPL y está disponible en la práctica totalidad de distribuciones Linux de propósito general. La empresa «MySQL AB» es la propietaria de este software y ofrece servicios a medida y soporte comercial para sus clientes.

Adicionalmente existe una versión de MySQL con licencia propietaria con mayor funcionalidad que su equivalente libre, a la que se puede acceder mediante el pago de la licencia de uso correspondiente.

Este modelo presenta multitud de ventajas a la empresa MySQL AB. Entre otras, tiene disponible toda una comunidad de desarrolladores interesados en implementar nuevas funcionalidades, solucionar fallos, reparar agujeros de seguridad, probar cada nueva versión, etc. Al ser MySQL. AB la propietaria de este código, tiene derecho a implementar las nuevas funcionalidades en la versión propietaria del software. Se consigue de hecho una relación simbiótica con la comunidad de software libre, en la que ambas partes, comunidad y empresa, salen beneficiadas del modelo de trabajo.

La comunidad de software libre obtiene:

- Un producto cada vez más desarrollado y completo
- Un código sobre el que poder investigar y mejorar

La empresa consigue:

- Una base enorme de implantaciones, llevando a una madurez muy rápida del software
- Un incremento en la generación de negocio (incremento en los contratos de soporte y de desarrollo de personalizaciones)
- Desarrollo de su software por terceras partes a una fracción del coste que supondría desarrollarlo internamente
- Nuevos usos para su software (MySQL AB nunca se hubiera imaginado que existiría un backend para Freenet de MySQL)

4.4 Desarrollo de componentes comerciales para productos de software libre

Este modelo es muy similar al anterior. Consiste en el desarrollo bajo licencia libre de aplicaciones específicas, generalmente de propósito general. Sobre estas aplicaciones se desarrollan determinados componentes comerciales que se distribuyen bajo el clásico modelo de licencias de uso que cubren determinadas necesidades específicas de un cliente.

4.4.1 Evolution

Un ejemplo de este modelo de negocio lo encontramos con el software Evolution de la empresa Ximian, recientemente adquirida por Novell.

«Evolution» es un cliente de correo y agenda de apariencia y funcionalidad similar a «Microsoft Outlook». Con una vida relativamente corta, esta aplicación ha llegado a ser considerada una de las mejores opciones como organizador personal.

Sobre este software, «Ximian» ha desarrollado un componente que se distribuye comercialmente capaz de integrarse con un servidor «Microsoft Exchange», de modo que sea posible para un usuario de Evolution la integración con redes Microsoft.

4.4.2 Kivio

Otro ejemplo lo podemos encontrar en la empresa «The Kompany», la cual ha desarrollado bajo licencia GPL un programa de diseño de diagramas denominado «Kivio». Kivio es un programa muy potente que en su versión GPL se ofrece con un conjunto de elementos gráficos muy reducido.

«The Kompany» ofrece conjuntos gráficos comercialmente, de modo que se potencie enormemente la funcionalidad de Kivio. También ofrece la herramienta de creación de conjuntos gráficos bajo licencia comercial.

4.5 Donaciones o suscripciones

Algunos negocios, sobre todo las publicaciones digitales basan su estructura de ingresos en las suscripciones de los usuarios. Este tipo de modelo de negocio por regla general está ligado a las revistas electrónicas, aunque también se observa en algunos proyectos de software libre.

4.5.1 Linux Weekly News (lwn.net)

Linux Weekly News es una revista electrónica en donde se publican todos los acontecimientos relacionados con el sistema operativo Linux que suceden. Su modo de trabajo consiste en la publicación de noticias diariamente, y la publicación de un resumen semanal de las noticias más importantes. Aunque LWN está presente en Internet desde hace bastantes años, el modelo de negocio que plantea no está todavía consolidado. Hace un año tuvo problemas financieros y estuvieron cerca

de cerrar. Es entonces cuando implantaron un sistema de generación de ingresos basado en suscripciones de los usuarios de modo que estos tuvieran acceso al boletín semanal y a determinadas noticias con una semana de antelación respecto al resto de los lectores. Tras una semana, estos artículos aparecen disponibles para todo el mundo. Este modelo de negocio está lejos de ser maduro, pero por lo menos ha permitido la supervivencia de LWN durante un año generando los ingresos necesarios para mantener el servicio que aportan a la comunidad.

4.5.2 Typo3

Typo3 es un gestor de contenidos liberado bajo licencia GPL y desarrollado en el lenguaje de programación PHP. El núcleo del gestor de contenidos está desarrollado por una única persona, que es la que dirige el desarrollo. Typo3 está diseñado de modo que el núcleo de la aplicación es el que ejecuta las tareas principales del gestor de contenidos (workflow, presentación, administración, etc) y la funcionalidad específica es implementada mediante plugins (módulo de foros, módulo de noticias, etc). Existen multitud de empresas desarrollando módulos para Typo3. El desarrollo del nucleo de Typo3 es guiado mediante donaciones, de modo que si una persona o empresa está interesado en acelerar una determinada funcionalidad del nucleo, puede enviar una donación al equipo de desarrollo de modo que este asigne mayor prioridad a dicha funcionalidad. Para que este modelo funcione correctamente es necesaria la disponibilidad de un software altamente maduro y con una gran base de usuarios.

4.5.3 Compiere

Compiere es un ERP (Enterprise Resource Planning) totalmente libre y con toda seguridad la opción más completa en cuanto a ERP's basados en software libre. Originariamente era una aplicación propietaria pero fue liberada cambiando su empresa a un modelo de servicios en vez de venta de licencias. El modelo de negocio de aplicación de servicios sobre un determinado software se comenta en otro capítulo y no incidiremos más en ello. Lo que si es relevante de Compiere en este momento son determinados aspectos de su modelo de desarrollo. Compiere utiliza Oracle como base de datos relacional, haciendo uso además de multitud de procedimientos almacenados que hacen altamente compleja la migración a otra base de datos que estuviera basada en Software Libre. Esto está creando importantes barreras de entrada para la implantación de este Compiere en pequeñas y medianas empresas. Parte de los ingresos de Compiere consisten en la distribución de un paquete que contiene una licencia de Oracle para una pequeña instalación así como soporte sobre la aplicación. La empresa por tanto no está muy interesada en desarrollar una capa de independencia de la base de datos que permita utilizar alguna base de datos libre como Postgresql.

Para el desarrollo de esta determinada necesidad planteada por la comunidad de software libre, la empresa ha estimado el coste de desarrollo y ha publicado la cantidad total de donaciones que requieren para abordar el proyecto. Una vez se alcance la cantidad en cuestión, se abordará el desarrollo de dicha funcionalidad. El resultado después de alrededor de dos años no ha sido muy positivo. Aunque existe una demanda muy importante, no ha sido posible recaudar ni siquiera una fracción de la cantidad requerida. Posiblemente en este caso la empresa que estuviera dispuesta a invertir en un proyecto de software libre también sería capaz de invertir en licencias Oracle para su implantación.

4.5.4 Transgaming y WineX

WineX es un proyecto similar a Crossover, que veremos más adelante. Mientras que Crossover está orientado a la ejecución de determinadas aplicaciones de oficina, WineX está orientado a la ejecución de diversos juegos diseñados para Win32 y sus librerías DirectX. WineX basa su modelo de negocio en el pago de suscripciones por parte de los socios. Estos tienen determinadas ventajas sobre los usuarios no socios, como la descarga de software adaptado a distribuciones estándar

4.6 Nuevos productos derivados de licencias tipo BSD

Existe una diferencia fundamental entre las dos grandes licencias libres, GPL y BSD. Mientras que con la licencia GPL todo el software derivado está obligado a llevar la misma licencia, con las licencias tipo BSD es posible re-licenciar el código derivado a cualquier tipo de licencia, incluidas licencias propietarias. Productos con licencia derivada de BSD podemos encontrar Apache, Wine o XFree.

4.6.1 Crossover Office / Plugin

Wine es un proyecto cuyo objetivo es la implementación de la API win32 sobre plataformas Intel. Históricamente se basaba en un derivado de la licencia BSD, aunque recientemente ha cambiado a una licencia tipo GPL. Crossover es una empresa que dispone de un producto capaz de ejecutar correctamente determinados programas desarrollados nativamente para Windows.

Entre estos programas se encuentra Microsoft Office, Internet Explorer, Cliente Lotus Notes, etc. El producto de Crossover está basado fundamentalmente en Wine, y sobre este han desarrollado las ampliaciones necesarias para ejecutar con garantías dichos programas. Esta empresa periódicamente envía sus desarrollos al proyecto Wine, de modo que estos se incorporan a la rama principal y pasan a ser mantenidos por la comunidad.

4.6.2 MacOS X

La última encarnación del sistema operativo MacOS tiene como base el núcleo UNIX de un sistema operativo bajo licencia BSD. Esto ha permitido a Apple la posibilidad de adaptación de dicho nucleo sin estar forzado a liberar el código fuente.

4.7 Integración de software

La integración de componentes o elementos de software no es un concepto nuevo. Desde hace muchos años el concepto de reutilización de código ha sido crítico en el éxito de una empresa de desarrollo. Lo que ha hecho el software libre es llevar la integración de software a límites mucho más extensos. El software libre por su naturaleza ofrece casi todas las ventajas para costruir un negocio basado en integración de software. Las bases de este tipo de negocio consisten en un potente I+D, capaz de conocer y analizar la mayoría de las herramientas disponibles basadas en software libre. De este conocimiento es posible construir aplicaciones a la medida del cliente mediante la integración de los elementos que más se adaptan a sus necesidades. La clave para este tipo de negocios es el desarrollo de software de integración evitando en la medida de lo posible la modificación de las aplicaciones a integrar y que permita evolucionar los distintos elementos de la solución final con el mínimo impacto sobre la solución alcanzada. Para las soluciones construidas en base a integración de distintas herramientas es vital la liberación del código desarrollado, sobre todo si la integración ha requerido la adaptación de alguna de las herramientas utilizadas.

4.7.1 openSistemas: integrador de soluciones

Un ejemplo de este modelo de negocio es «openSistemas» (www.opensistemas.com), empresa española que ha construido un modelo de negocio alrededor de la integración de aplicaciones basadas en software libre, ofreciendo soluciones «llave en mano» e implantaciones de productos paquetizados en entornos de cliente. La clave del modelo de negocio es un potente I+D, fundamentalmente dedicado al análisis de aplicaciones existentes en el mundo del software libre desde la óptica de la integración, dirigido desde el departamento comercial. La solución a las necesidades del cliente se construye generalmente mediante las siguientes fases:

- Análisis de las necesidades del cliente
- Identificación de los grandes sistemas que integrarán la solución
- Identificación de las necesidades de desarrollo
- Desarrollo de elementos necesarios
- Integración de la solución mediante utilización de estándares LDAP, SGBD, etc

La integración se realiza de forma limpia, evitando modificar el código de las distintas aplicaciones a integrar. De este modo es posible beneficiarse del proceso de revisión y actualización de versiones de cada proyecto individual basado en software libre sin realizar esfuerzos importantes en reintegrar las modificaciones. En casos en los que es necesario realizar modificaciones a algún proyecto original, dichas modificaciones deben ser siempre publicadas y enviadas al proyecto principal, de modo que se garantice la compatibilidad con la solución global ante futuras revisiones del código.

5-Conclusiones

Hasta hace unos pocos años no se había podido generalizar la viabilidad del software libre como modelo de negocio. Por regla general, las buenas ideas fallaban por falta de masa crítica de mercado, falta de soporte de grandes empresas u otros motivos. En los últimos años, la creciente competitividad, el aumento de madurez del mercado tecnológico, el propio avance del Software Libre y su probada eficiencia y calidad junto con una situación económicamente débil, han provocado el comienzo de la asunción de un cambio de paradigma en el desarrollo y distribución de software. Los clientes empiezan a no estar satisfechos con la adquisición de productos, sino que quieren una personalización, una adaptación a sus problemas reales. De esta forma muchas empresas han pasado de un modelo basado en venta de producto a otro basado en venta de servicios y soporte asociados. Es en este nuevo modelo donde el Software Libre es casi imbatible. El cambio que se está produciendo tiene además implicaciones de orden político que no escapan a los gobiernos y grandes empresas y que provocan movimientos muy importantes impensables hace muy poco tiempo. Por ejemplo en España estamos viendo cómo el uso del software libre se utiliza en los programas electorales como un arma de diferenciación política.

Utilizando soluciones basadas en software libre no es necesario estar constantemente reinventando la rueda. Si nos paramos a pensar un poco es un modelo muchísimo más lógico desde el punto de vista del desarrollo de software. Los nuevos esfuerzos parten de código ya existente y disponible garantizando que los nuevos esfuerzos no tienen que partir de cero. Desde mi punto de vista, consiste en la implementación del método científico al desarrollo de software. Posiblemente, la consolidación de los modelos de negocio basados en software libre estaba pendiente del alcance de la masa crítica de usuarios necesaria para convertir el planteamiento idealista basado en el concepto de «hacker» en planteamientos de negocio mucho más sólidos que podemos encontrar en la actualidad.

Sobre el autor

Fernando Monera (fmonera@opensistemas.com), Ingeniero Informático, MBA en ICAI, trabaja en la actualidad como director general y socio fundador de openSistemas (http://www.opensistemas.com), empresa focalizada en integración de soluciones basadas en software libre.
Su relación con el software libre comenzó en 1997 interesado fundamentalmente por el aspecto sociológico detrás del concepto de software libre, representado en gran medida en el artículo «La catedral y el Bazar» de Eric S. Raymond.
En 2001 fundó un portal de supercomputación de habla hispana denominado hispaCluster (http://www.hispacluster.org) en pleno funcionamiento hoy en día y patrocinado por openSistemas.
Desde 1997 hasta hoy ha estado relacionado con el software libre tanto laboral como personalmente, involucrado en diferentes proyectos como TWIG (herramienta groupware basada en plataforma web) o Jensen (linux para plataformas Alpha), fomentando el uso de software libre en todos los ámbitos que le ha sido posible.

Congreso Hispalinux

Durante septiembre 2003 tuvo lugar, en la Universidad Rey Juan Carlos de Madrid, la VI Edición del Congreso Hispalinux. Se trata de uno de los eventos más destacados dentro del ámbito del software libre, en el que se han dado cita los responsables de grandes proyectos internacionales, instituciones educativas, hackers y profesionales de las nuevas tecnologías, y en el que openSistemas ha participado de manera activa. Fernando Monera, en representación de OenSistemas (www.opensistemas.com), formaba parte del elenco de ponentes invitados al Congreso. Reproducimos aquí su conferencia, bajo el título de "Modelos de negocio basados en software libre".



Scripting en Microsoft Windows 2000-2003

En este artículo daremos una introducción sobre las tecnologías de scripting incluidas en el sistema operativo de Microsoft Windows 2000-2003. Hablaremos sobre los conceptos básicos, daremos algunos ejemplos y comentaremos sobre las diferentes opciones que nos provee Microsoft : Visual Basic Scripting Edition (VBScript), Windows Script Host (WSH), Active Directory Service Interfaces (ADSI), and Windows Management Instrumentation (WMI).

Microsoft provee en la web :"The Scripting Guide", http://www.microsoft.com/resources/documentation/windows/2000/server/scriptguide/en-us/default.mspx (del cual hemos extractado la mayor parte de lo aquí presentado) que posee cientos de scripts pre-escritos que le permitirán realizar muchísimas tareas administrativas: hacer backups y limpiar event logs; monitorear uso de recursos por los diferentes procesos, lectura y escritura al registry; y manejo de cuentas de usuarios, cuentas de computadoras, impresoras y servicios. Este mismo material se encuentra en el excelente libro de Microsoft Press: Microsoft Windows 2000 Scripting Guide.

Introducción

Las computadoras y las redes de computadoras continúan haciéndose mas poderosas y complejas lo que hace que los administradores de sistemas deban enfrentar a diario nuevos desafíos. No hace mucho, ellos debían manejar no mas de una docena de maquinas (la mayoría localizadas dentro de la misma área geográfica), un número reducido de cuentas de usuarios, y unas pocas aplicaciones que no tenían idea de la existencia o necesidad de una red local.

Hoy, todo ha cambiado y se deben enfrentar a cientos de máquinas o aún miles con un numero muy grande de usuarios. Estas computadoras situadas quizás muy distantes unas de otras ejecutarán cientos de aplicaciones y dependerán de recursos de la red incluyendo servidores de archivos, servidores DHCP (Dynamic Host Configuration Protocol) y servidores Domain Name System (DNS), e impresoras compartidas. Esto ha hecho que los administradores de sistemas deban encontrar herramientas de manejo ("management tools") que tengan los siguientes requisitos:

- Puedan ser modificadas para obtener los requerimientos particulares a una dada organización
- Puedan operar sobre múltiples computadoras y computadoras remotas
- Sean rápidas, fáciles de implementar y de reducido costo.

Los scripts nos proveen tales requerimientos y más. Además, scripting no adiciona costo alguno en hardware o software mas allá de tener Windows 2000-2003 como sistemas operativo.

Conceptos de lenguajes y tecnologías de scripting para administradores de sistemas.

Para poder escribir scripts para poder manejar computadoras bajo windows, necesitará conocer a fondo las tecnologias de scripting que se incluyen en Windows 2000-2003. Las posibilidades que se le presentan son: Microsoft® Visual Basic® Scripting Edition (VBScript); Windows Script Host (WSH); the Script Runtime library; Active Directory Service Interfaces (ADSI); and Windows Management Instrumentation (WMI).

Hasta hace muy poco scripting era algo que el administrador de sistemas no hacía. En general se tiene la siguiente idea de scripting: es difícil; scripting lleva mucho tiempo; nos obliga a aprender todo tipo de jergas y a manejar una cantidad enorme de acrónimos: - WSH, WMI, ADSI, CDO, ADO, COM. Los administradores de sistemas no tienen el tiempo ni el background para poder ser escritores de scripts.

Es realmente así? Uno de los propósitos de este artículo es responder algunas de estas preguntas.

Primer pregunta : scripting es difícil?

Analicemos el siguiente script que veremos realiza una tarea que puede ser muy útil para el administrador:

```
Set objNetwork =
CreateObject("WScript.Network")
objNetwork.MapNetworkDrive "X:", "\\atl-fs-01\public"
```

Aunque Ud no tenga la menor idea de scripting y aunque este perplejo por el anterior, seguramente podrá adivinar que este script mapea el drive X al recurso compartido \\atl-fs-01\public. Y eso es justamente lo que hace. Si Ud es un administrador de sistemas, es decir si sabe lo que es mapear un drive y el concepto de un recurso compartido y los paths de la Universal Naming Convention (UNC) el salto de hacer esto vía una graphical user interface (GUI) o línea de comandos vía un script no es difícil.

Nota

Si UD esta confundido sobre lo que quiere decir scripting pienselo en estos términos: Alguna vez se encontró tipeando una y otra vez el mismo comando para hacer una tarea? Alguna vez se encontró clickeando los mismos botones en la misma secuencia, en el mismo wizard de modo de completar la misma acción. Y aun mas debió repetir el mismo proceso, para digamos, múltiples computadoras y cuentas?

Los scripts le permitirán eliminar parte de este trabajo repetitivo. Un script es un archivo que UD crea que describe los pasos requeridos para realizar una tarea. Después de crear el script lo podra "correr" ("run") una y otra vez ahorrándole tiempo y esfuerzo. Necesitara crear el script una sola vez y luego re-usarlo cada vez que deba repetir la tarea.

Hay que admitir que no todos los scripts son tan simples como el que acabamos de ver. Pero si mira ejemplos de scripts vera que son pequeños no superando 15 a 20 líneas y en muchos de ellos entenderá que realizan independientemente del grado de conocimientos de scripting que tenga.

Lleva mucho tiempo hacer un script?: Puede ser, dependiendo de lo complejo que se haga. Es importante balancear el tiempo invertido en hacer el script y el tiempo que nos ahorrara al usarlo.

Aquí le mostramos otro ejemplo de un script para realizar backups y además limpiar todos los "event logs" de la computadora:

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:" _
    & "{impersonationLevel=impersonate, (Backup, Security)}!\\" _
        & strComputer & "\root\cimv2")
Set colLogFiles = objWMIService.ExecQuery _
    ("Select * from Win32_NTEventLogFile")
For Each objLogfile in colLogFiles
    strBackupLog = objLogFile.BackupEventLog _
        ("c:\scripts\" & objLogFile.LogFileName & ".evt")
    objLogFile.ClearEventLog()
Next
```

Ya este script no es tan intuitivo como el anterior. Aun más, para escribir un script como este deberá conocer un poco más de scripting que lo básico y en particular si usamos como en este caso Windows Management Instrumentation WMI.

Y además deberemos tipear en Microsoft® Notepad las 11 líneas. Esto puede llevar tiempo. Pero piénselo de este modo: cuanto tiempo le lleva realizar la misma tarea?. Siendo además una tarea rutinaria y tediosa muy probablemente comenzara a NO realizarla tan seguido.

Con un script UD realizará la tarea en minutos. Y si lo sofistica podrá hacer la tarea en todas sus maquinas. Más aun quizás comience a programar los scripts. Ya vera como paga la inversión.

Pero uno deberá aprender como programarlos? Si, si queremos ser expertos en scripting.

Pero considere el siguiente script que nos retorna los nombres de todos los servicios instalados en nuestra computadora:

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:" & _
"{impersonationLevel=Impersonate}!\\" &
strComputer & "\root\cimv2")
Set colItems = objWMIService.ExecQuery("Select * from Win32_Service")
For Each objItem in colItems
    Wscript.Echo objItem.Name
Next
```

Parece bastante complejo. Realiza lo siguiente:

- Usa Automation object methods y properties
- Usa elementos de Microsoft® Visual Basic® Scripting Edition (VBScript) tales como loops tipo "For Each" para iterar a través de los elementos de una colección.
- Requiere de COM ((Common Object Model) moniker.
- Usa WMI object paths, namespaces, y classes Ejecuta un query a un string escrito en WMI Query Language

Es bastante frustrante necesitar conocer tanto para realizar un script de 7 líneas. Por eso se cree que es complicado.

Pero la realidad es que uno no debe conocer tan a fondo COM y Automation para escribirlo.

Pero supongamos que ahora queremos hacer un script que nos devuelva los nombres de todos los procesos que corren en nuestra computadora en lugar de los nombres de los servicios. El script seria como:

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:" & _
"{impersonationLevel=Impersonate}!\\" &
strComputer & "\root\cimv2")
Set colItems = objWMIService.ExecQuery("Select * from Win32_Process")
For Each objItem in colItems
    Wscript.Echo objItem.Name
Next
```

Pero porque este script es tan especial? En realidad no lo es. Y ese es el punto. Mire el detalle de "Win32_Process". Esta es la única parte del scirpt de procesos que difiere del script de servicios. Sabe UD algo mas sobre COM monikers o WMI object paths que hace un minuto. Probablemente NO. Sin embargo pudo modificar un template básico para obtener lo que deseaba. Quisiese obtener el nombre de la tarjeta de video instalada en su computadora? Pruebe el siguiente script:

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:" & _
"{impersonationLevel=Impersonate}!\\" &
strComputer & "\root\cimv2")
Set colItems = objWMIService.ExecQuery("Select * from Win32_VideoController")
For Each objItem in colItems
    Wscript.Echo objItem.Name
Next
```

Es siempre tan facil?. No, no lo es. Y en los argumentos anteriores hemos simplificado algunas cosas. Como sé que debo tipear "Win32_VideoController" y no "Win32_VideoCard"?. Y que sucede si quisiésemos más información que solo el nombre de la tarjeta de video. Lo que queremos remarcar no es que uno puede comenzar a escribir scripts sin saber NADA,sino que lo podrá hacer sin necesidad de saber TODO. Si desea poder manejar cómodamente COM Monikers y WMI object paths antes de comenzar esta muy bien. La otra opción es tomar scripts ya hechos y modificarlos. Luego podremos volver y volvernos expertos. De esta segunda manera comenzamos en forma inmediata.

¿Porque es que scripting obtuvo tan mala reputación en el ámbito de los administradores de sistemas?

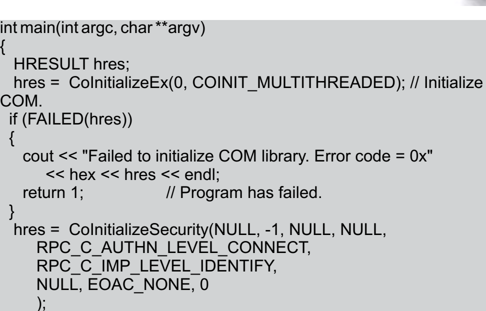
Si scripting es tan sencillo, porque es que tiene una reputación de ser muy complejo? Y si es tan útil, porque no está difundido en el mundo de los administradores?. Es difícil entender porque ellos le darían la espalda a algo que puede llegar a serles tan útil. Puede haber varias razones pero probablemente todo se remonta al nacimiento de Microsoft® Windows® Script Technologies. Tanto VBScript y Microsoft® JScript® (los



dos lenguajes de scripting incluidos en los sistemas operativos de Microsoft) comenzaron como un modo de agregar scripting del lado del cliente a las paginas web. Esto estaba muy bien con los desarrolladores de Internet, pero era de poco uso para los administradores de sistemas. Como resultado a scripting se lo asocio con desarrollo de paginas web. (aun hoy muchos de los ejemplos de la documentación oficial de Microsoft muestra código embebido en paginas web).

Mas tarde, nació Windows Script Host (WSH). WSH proveyó un modo en que lenguajes de scripting y tecnologías de scripting fuesen usados fuera de Internet Explorer. De hecho WSH estuvo dirigido desde el vamos a los administradores. Aun así, scripting no prendió en ese mundo.

Inicialmente esto pudo deberse a la falta de documentación y posicionamiento. Era difícil hallar información sobre como usar VBScript o JScript como una herramienta para los administradores. Era casi imposible hallar documentación sobre tecnologías como WMI o Active Directory Service Interfaces (ADSI). Aun cuando estas tecnologías fueron documentadas (tipicamente en kits de desarrollo de software) la documentación estaba dirigida a los desarrolladores. De hecho, ejemplos de código eran escritos en C++, más que en un lenguaje de scripting. Por ejemplo, suponga ser un administrador típico. (con mucho conocimiento de Windows y mínimos conocimientos de programación). Y suponga haber buscado "scripting" en la pagina web de Microsoft y se encontró con un ejemplo como este.



```
int main(int argc, char **argv)
{
   HRESULT hres;
   hres =  CoInitializeEx(0, COINIT_MULTITHREADED); // Initialize COM.
  if (FAILED(hres))
  {
     cout << "Failed to initialize COM library. Error code = 0x"
          << hex << hres << endl;
     return 1;                 // Program has failed.
  }
  hres =  CoInitializeSecurity(NULL, -1, NULL, NULL,
       RPC_C_AUTHN_LEVEL_CONNECT,
       RPC_C_IMP_LEVEL_IDENTIFY,
       NULL, EOAC_NONE, 0
       );
```

No hay que ni decirlo, pero muy pocos administradores verían a WMI o ADSI como una herramienta de utilidad.

Hoy, por supuesto no hay falta de documentación de scripting.

Una búsqueda en empresas de venta de libros en Internet hallaría mas e 350 títulos bajo la palabra: "VBScript".

Esto es bueno. Pero aun, algunos libros enfocarían scripting como una herramienta para desarrolladores o se enfocarían exclusivamente en VBScript y WSH.

Aunque VBScript y WSH sean importantes tecnologías de scripting, ellas por si solas no le alcanzaran para realizar tareas de scripting a nivel administrador.

Muy pocos de esos libros apuntarán al administrador de sistemas y muy pocos cubren tecnologías muy fundamentales como WMI y ADSI en profundidad.

Un administrador de sistemas que escogiese uno o dos libros de scripting al azar no entendería porque esta tecnología le serviría para administrar sus computadoras bajo Windows.

El libro: *Microsoft® Windows® 2000 Scripting Guide*

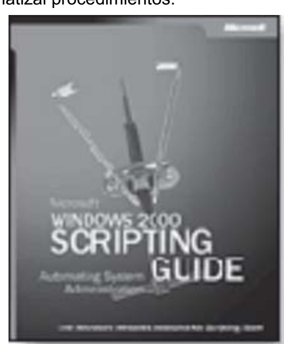
Recomendamos a aquel que desee adentrarse en scripting para el entorno de sistemas operativos Windows de Microsoft que utilice este libro como primer paso. Que lo hace diferente en su modo de ver scripting?. Tiene por lo menos cuatro rezones bien definidas:

El foco es en scripting desde el punto de vista de un administrador de sistemas.

El libro esta centrado en tasks (tareas) mas que alrededor de scripting mismo

Combina tutoriales con elementos prácticos

En este libro se reconoce que cada vez mas es indispensable para las organizaciones automatizar procedimientos.



Una nota sobre VBScript

Todos los scripts presentados en el libro *Microsoft® Windows® 2000 Scripting Guide* usan VBScript.

La decisión se debió a varias razones que detallamos a continuación ya que nos sirven para poner en contexto los diferentes lenguajes de scripting mas populares.

Con la excepción de Perl, VBScript es el lenguaje mas popular para escribir scripts para el administrador de sistemas.

A diferencia de Perl, VBScript (junto a Jscript) esta instalado automáticamente en todos los Windows 2000 en adelante.

No hay nada que comprar o nada que instalar.

VBScript es mas sencillo de aprender que Jscript. Además es muy parecido a Visual Basic (un lenguaje de programación que muchos administradores ya conocen).

Conclusión

Scripting es hoy una herramienta necesario para la administración de sistemas.

Como aprenderlo dependerá de nuestro gusto. Podemos volvernos expertos en los lenguajes para luego comenzar o como se enfoca en el libro "Microsoft 2000Scripting guide" podremos utilizar scripts ya hechos e ir modificándolos. .Recalquemos que de algún modo el lenguaje de scripting que utilicemos es irrelevante. Por si solo VBScript ofrece poco soporte para la administración de sistemas.

Complementado con WSH, WMI, ADSI, y otras tecnologías de scripting se transforma en muy útil.

LAS COMPUTADORAS QUE SOPORTARON LA PRUEBA DEL TIEMPO

A pesar de las predicciones que decían que las PCs iban a reemplazarlas, IBM sigue vendiendo mainframes 40 años después de su lanzamiento.

En una casa del siglo XIX en Hampshire, un pequeño grupo de científicos trabajaron por cuatro años en un producto que cambiaría el mundo de negocios para siempre. Afuera descansan varios acres de maderas y jardines; adentro, una confusión de cables, transistores y una variedad de elementos electrónicos en medio de cabezas de ciervos que recubrían las paredes. Corría 1964 y los investigadores estaban inventando lo que se convertiría en las primeras mainframes de "Internacional Business Machines" (IBM).

"La atmósfera era fantástica. No había prácticamente nadie con más de 30 años. El lugar estaba vivo con gente brillante. No había burocracia. No sabíamos que había cosas que no se podían hacer, así que simplemente las hacíamos. Trabajábamos a todas horas. Pero era tan emocionante", recuerda Tony Proudman, de entonces 30 años, que estaba involucrado en el proyecto original en Hursley Park, Hampshire. Otros cuatro laboratorios de IBM también tomaron parte en el desarrollo pero Proudman recuerda que el proyecto de Hursley fue el primero en obtener componentes en operación.

Había una percepción entre la gente de IBM que se estaba poniendo en juego el futuro de la compañía con esta tecnología.

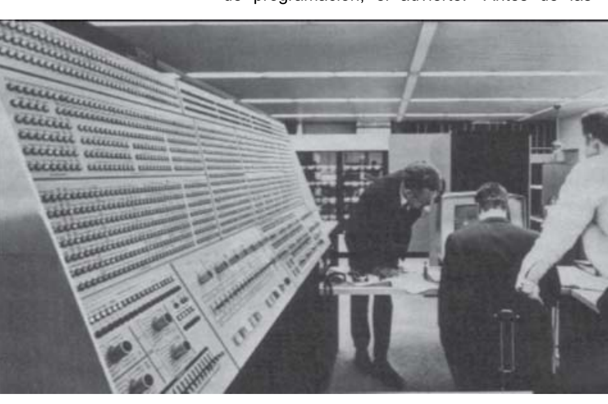
La mainframe S/360 celebró su cumpleaños número 40 el 7 de Abril de este año. Llevó cuatro años de construcción, en cinco localidades, incluyendo Hursley, Boeblingen en Alemania, y tres laboratorios en EE.UU, y costó 5 mil millones de dólares en investigación y desarrollo, en un momento en el que el ingreso de las compañías era alrededor de los 2.200 millones.

"Había una percepción entre las personas de IBM de que verdaderamente estaban apostando su empresa en tecnología", dice Doug Neilson, un veterano de IBM con 28 años en la empresa, que se encontró por primera vez con mainframes como cliente en 1968.

Rekesh Kumar, vicepresidente de Analistas de Investigación de Tecnología del Grupo Meta, agrega: "Hubo muchísimo conflictos internos en cuanto a la estrategia. Muchas personas pelearon en contra de esto. Y había un gran riesgo adjunto, con una pequeña garantía de que funcionaría. Nadie más pensaba de esta forma (sobre computación)."

La afirmación de IBM de haber inventado las mainframes, es discutible. "Nosotros tenemos mainframes desde principios de la década del 50", protesta Don Scoggin, vicepresidente de marketing en Unisys. Él cita las elecciones presidenciales de 1952 en Estados Unidos, cuando CBS usaba una máquina del predecesor de Unisys, Univac, para predecir el resultado con uno por ciento de margen de error.

Es una cuestión de definiciones. Dice Kumar: "Antes, había máquinas individuales con sus aplicaciones propias y dedicadas, y sistemas operativos, pero las mainframes permitieron una carga de trabajo múltiple, cada uno corriendo al mismo tiempo en una misma máquina, en vez de en 20 o 30 computadoras individuales, cada una haciendo cosas diferentes. Y las mainframes podían hacerlo mucho más velozmente". Bajo esta definición, discute que IBM fue el primero.



Scoggin argumenta con detalles sobre las Burrough B5000 (Bourroughs fue otro predecesor de lo que se convertiría en Unisys), que con dos procesadores realizaba multi-programming, multi-tasking y memoria virtual en 1963.

Quien quiera que haya sido el primero en alcanzar este hito memorable, fue IBM el más identificado con la tecnología mainframe. Rápidamente llegaron los competidores, incluyendo Amdahl, creada por un ex empleado de IBM, Hitachi, Siemens y Bull. Hoy en día estas compañías venden unas pocas unidades al año, con el volumen de ventas viniendo de IBM. Se calcula que esta tecnología ha generado 60.000 puestos de trabajo desde su comienzo

Con el advenimiento de las mainframes, veintenas de grandes compañías reorganizaron sus actividades computacionales. En las décadas del 60 y 70, las mainframes eran usadas para automatizar tareas clericales. Las compañías eran capaces de procesar datos en sus informes y planillas, luego empezar a integrar bases de datos jerárquicas y relacionales, lo cual antes era imposible.

"Las mainframes verdaderamente transformaron para siempre la forma en que los negocios utilizan las computadoras, ya que no se tenían que usar diferentes sistemas pequeños y agruparlas en clusters de máquinas. Podrían comprar máquinas grandes con poder de procesamiento dedicado. Si se necesitaba ampliar se lo hacía sobre la misma máquina. Requería un conjunto de ingenieros, procesadores y sistemas operativos completamente nuevos, todas las cosas que hoy en día damos por sentado", explica Kumar.

Las mainframes también transformaron software de programación, él advierte: "Antes de las mainframes, el código de proceso de transacción tenía que ser escrito individualmente. Pero las mainframes tenían (trozos de código de programación) que se podían volver a usar, y de esta manera solo habia que modificar un poco para algunas aplicaciones según sea necesario. Esto hizo que todo el ciclo de vida de programación sea mucho más rápido. Creó un nuevo ecosistema de programadores y lenguajes que no había existido hasta entonces. En los últimos años, esto cambió el panorama de la computación."

Los primeros modelos de producción de IBM fueron a organizaciones tales como Bank of America, Time Life, Allstate Insurance, Nasa y Westminster Bank.

Ocupaban 30 o 40 metros cuadrados y las disponían como racks electrónicos alrededor de un gran marco (frame) de metal, con sus "frames" subsidiarios por esto el término "mainframe" (frame principal) para el sistema central de procesamiento. Hoy en día, una máquina equivalente a esta ocuparía 1 metro cuadrado pero tendría un poder de procesamiento un millón de veces mayor.

Asimismo, el poder de procesamiento se ha incrementado tanto en cuatro décadas que si en una de las máquinas originales S/360 se hubiese dejado corriendo constantemente un proceso desde este lanzamiento, la cantidad de procesamiento que hubiera realizado sería equivalente a lo que realizaría en sólo seis horas una de las mainframes de hoy en día.

Que hoy, un equivalente de mainframe de aquella época todavía se comercialice ha sido una sorpresa para los que anunciaron con confianza la muerte de las mainframes en los años 90, momento en el que los ordenadores personales habían llegado a dominar los escritorios de las oficinas y los servidores más pequeños ocuparon los cuartos del fondo.

Las PCs y los servidores, cubrirían el procesamiento, absorberían a las mainframes ya que son más versátiles y podrían acoplarse para ofrecer juntos un poder similar de procesamiento. Pero esa muerte no llegó. "Usted también querría una muerte como esta, déjeme decirle", dice Neilson. La tecnología de 40 años está disfrutando un nuevo impulso ya que las compañías buscan correr sus propias operaciones de e-business de una forma segura y eficiente en lo que todavía se conoce cariñosamente como "big iron".

Kumar explica que PCs actuando en conjunto no es lo mismo que una mainframe: "Uno puede transportar a la gente que llena un estadio de football en bicicletas, pero es más fácil si lo hace con autobuses." Otras ventajas que incluyen son confiabilidad, seguridad, gran capacidad para tareas complejas y lo que es conocido como accesibilidad - los mainframes generalmente no necesitan ser detenidas para realizarles para mantenimiento.

Los mainframes pueden lidiar con millones de usuarios y hacer diez mil transacciones en un segundo. Por el otro lado son caras y requieren mantenimiento especializado y los clientes pueden encontrarse sujetos a un solo proveedor.

Mainframes todavía son muy usados por las grandes compañías. IDS estima que alrededor de 3000 son entregadas cada año, de las cuales aproximadamente 2700 son de IBM. Este número ha permanecido estable por varios años, dice Chris Ingle, un consultor del grupo IDC, con el año más pobre en el 2002, en el cual IBM obtuvo ganancias de U$4bn por sus mainframes, creciendo a $4.2 bn en el 2003. Hoy en día los mainframes pueden costar entre $200,000 y

$20m. Son usadas en aplicaciones como ser actividades bancarias, correduría (brokerage), reservaciones de aerolíneas y sistemas del gubernamentales. En particular a lo que hace a seguridad.
Ciertas tecnologías que antes se realizaban en las mainframes, como ser sistemas de recursos de administración que permiten que su carga sea distribuida a diferentes procesadores fueron pasados a maquinas mas livianas como por ejemplo servidores Intel, mientras que tecnologías de servidores han sido aplicadas hacia mainframes. Por ejemplo, IBM está comenzando a utilizar el sistema operativo gratuito, Linux, en los mainframes, junto con



programas populares de lenguaje como Java y C.
Ingle dice:"Existe como una idea de no reemplazar las mainframes o hacer algo con ellos, como se habló hace algunos años atrás. Es un producto caro, pero hace su trabajo y no hay nada que pueda tomar su lugar. Muchas compañías tenían proyectos para reemplazarlas, pero luego se dieron cuenta de que era muy riesgoso e inestable. No ha habido ni una caída en las ventas yse predice ventas estables para los próximos años".
Para enfatizar la durabilidad de una mainframe, IBM anunciará un nuevo modelo, el Z890. Este será uno de las mainframes más pequeños de la compañía, y puede ser up-gradeada por etapas si es necesario. También contiene nueva tecnología que optimiza la máquina para usarse con el popular programa de lenguaje Java. Neilson dice: "Estamos haciendo esto porque vemos el rol del mainframe en el corazón del e-business. Estamos optimistas en cuanto al futuro".

NUEVOS ANTIVIRUS

El software se denomina Kaspersky Anti-Virus para Linux/UNIX mail Servers V 5.0, y ofrece nuevas características, incluyendo la posibilidad de comprobar todos los datos de los correos electrónicos en tiempo real.
Con esta función, el programa permite buscar y neutralizar los códigos maliciosos que encuentre en el servidor, asegurando que sólo los datos confirmados están seguros.
El sistema soporta FreeBSD, Open BSD y versiones populares de Linux.
La versión 5.0 permite a los administradores de sistemas modificar determinados requerimientos. En esta versión, los archivos se han unificado en una configuración simple, lo que permite a los administradores de sistemas controlar el producto desde una única consola.
Los usuarios de la versión anterior -la cuarta- podrán descargarse gratuitamente la versión quinta desde el sitio web del fabricante.
www.kaspersky.com

FERRET CMS, LIBERADO

La versión 0.4 de Ferret CMS (www.opensistemas.com/Ferret_CMS.624.0html), ha sido liberada por openSistemas bajo licencia GPL, lo que supone un importante avance para la comunidad de Software Libre pero, sobre todo, constituye un gran paso en el desarrollo tecnológico de las PYMES, ya que está pensado para satisfacer las necesidades de muchos portales corporativos. Ferret CMS es un Gestor de Contenidos sencillo, rápido y potente para crear sitios web de pequeño y mediano tamaño y posee una ventaja primordial sobre otros gestores de contenidos disponibles: el interfaz de administración es sencillo y fácil de usar y su configuración reúne las mismas características. Además, con Ferret es posible crear un portal completamente funcional en un tiempo récord.
Resulta manejable para un usuario no técnico y ofrece la máxima facilidad en su instalación y administración, otorgando un alto grado de flexibilidad al desarrollador.

¿Qué pérfil de técnico en IT necesita España?

El área de Selección de TEA CEGOS, firma de consultoría, formación y selección, ha elaborado un estudio sobre la demanda de empleo en el sector de informática y nuevas tecnologías, Y según este estudio, los profesionales más demandados, por orden de necesidad son:

de Sistemas (Unix, WNT-2000, Linux...)- Redes Seguridad y correo electrónico."
"Analistas y Desarrollo en C, C++, Java, Vb, Delphi, …"
"Analistas y Desarrollo Web (html, java, sql, asp, php, "vigente" …)- Redes tcp/ip
"Consultor Analista Programador o Administrador de aplicaciones ERP's"
"Diseñador Gráfico Infografista Experto en contenidos Web"
Ingenieros en Telecomunicaciones Planificación de Red comunicaciones móviles fijas cable ict…"
"Soporte a usuarios postventa aplicaciones help desk".

PKI

Conceptos básicos

En este artículo desarrollaremos los conceptos de la llamada infraestructura de claves públicas (Public Key Infrastructure, PKI).

Introducción

La criptografía protege a los usuarios ya que permite la encriptación de datos y autenticación de usuarios. Esta tecnología permite al receptor de un mensaje electrónico comprobar la identidad del emisor *(autenticación)*, asegura que un mensaje sólo puede leerlo la persona a la que va dirigido *(confidencialidad)* y asegura al destinatario que un mensaje no ha sufrido ninguna alteración durante su envío *(integridad)*. En NEX 5 y 6 ya describimos los conceptos básicos de criptografía: cifrado de claves simétricas, el cifrado de claves públicas, los algoritmos hash, las firmas digitales y el intercambio de claves.

Una Infraestructura de Claves Públicas (PKI) engloba las técnicas y los estándares usados para administrar certificados y claves públicas y privadas de modo de poder autenticar la validez de cada una de las partes implicadas en una transacción electrónica. En lo que sigue desarrollaremos los conceptos de PKI: certificados, entidades emisoras de certificados (Certificate Authorities, CA), jerarquías de entidades raíz, registración, inscripción de certificados, revocación de certificados y validación de la cadena de certificados.

Comprender los conceptos de criptografía y PKI es la piedra fundamental para el entendimiento de muchísimos conceptos relacionados a la seguridad en IT. Antes de comenzar es muy importante que se distinga entre criptografía de claves públicas y PKI. El primer concepto (criptografía de claves públicas) ya fue desarrollado en dos artículos anteriores de NEX ("Elementos de criptografía" en los "Suplemento de Seguridad" en NEX 5 y 6) junto con los conceptos de criptografía de clave secreta, firma digital e intercambio de llaves secretas. Si UD. no esta familiarizado con éstos le recomendamos leerlos antes de continuar. Una vez leído el presente artículo UD. podrá abordar artículos más técnicos que utilicen PKI. PKI es un tópico muy extenso y necesitaríamos un libro entero para poder desarrollarlo. Aquí solo daremos una introducción.

El término infraestructura de claves públicas (PKI) se utiliza para describir las directivas, los estándares y el software que regulan o manipulan los certificados y las claves públicas y privadas. En la práctica, PKI hace referencia a un sistema de certificados digitales, entidades emisoras de certificados (CAs) y otras entidades de registro que comprueban y autentican la validez de cada parte implicada en una transacción electrónica. Los estándares de la infraestructura de claves públicas siguen evolucionando, aunque se estén implementando de forma generalizada como elemento necesario del comercio electrónico.

Lo que sigue nos ayudará a comprender qué es una PKI y a conocer los servicios necesarios para construir una. Diferentes conceptos relacionados con la PKI se explican en las subsecciones siguientes:

- Certificado
- Entidad emisora de certificados (Certification Authority, CA)
- Jerarquías de entidades raíz
- Registración
- Inscripción de certificados
- Revocación de certificados
- Validación de la cadena de certificados

Certificado

Un certificado de claves públicas, a menudo denominado simplemente certificado, se utiliza para autenticar y asegurar el intercambio de información en Internet, extranets e intranets. Al emisor y firmante del certificado se le conoce como entidad emisora de certificados (Certification Authority, CA), que se describe en la siguiente sección. La entidad a la que se le emite el certificado es el "*tema (subject)*" del certificado.

Un certificado de claves públicas es una declaración firmada digitalmente que vincula el valor de una clave pública a la identidad del "*subject*" (persona, dispositivo o servicio) que posee la clave privada correspondiente. Al firmar el certificado, la entidad emisora de certificados (CA) atestigua que la clave privada asociada a la clave pública del certificado está en posesión del "*subject*" indicado en el certificado.

Los certificados pueden emitirse con objetivos diferentes, entre ellos la autenticación de usuarios Web, la autenticación de servidores Web, la seguridad del correo electrónico con Secure/Multipurpose Internet Mail Extensions (S/MIME), IP Security (IPSec), Secure Sockets Layer/Transaction Layer Security (SSL/TLS) y firma de código. Si por ejemplo en una organización se utiliza la entidad emisora de certificados empresarial de Windows 2000, los certificados podrán utilizarse para iniciar sesión en dominios de Windows 2000. Los certificados también pueden emitirse de una entidad emisora de certificados (CA) a otra con el fin de establecer una jerarquía de certificados.

En el certificado, el *subject* puede especificarse con varios nombres. Como nombre principal de usuario (para certificados de usuarios finales), nombre de directorio, nombre de correo electrónico, nombre DNS, etc. Los certificados también contienen la siguiente información:

- El período de tiempo durante el cual el certificado es válido.
- El número de serie del certificado, que la entidad emisora de certificados (CA) garantiza como único.
- El nombre de la entidad que emitió el certificado y la clave utilizada para firmarlo.
- Un identificador de la directiva seguida por la entidad emisora de certificados (CA) para autenticar que el *subject* es quien dice ser (más adelante se ofrecerán detalles sobre la directiva de la entidad emisora de certificados).
- Los usos que se darán al par de claves (la clave pública y su clave privada asociada) identificadas en el certificado.
- La ubicación de la Lista de revocación de certificados (Certificate Revocation List, CRL). La CRL es un documento mantenido y publicado por una entidad emisora de certificados (CA) que enumera aquellos que se han revocado. Una CRL se firma con la clave privada de la entidad emisora de certificados (CA) para garantizar su integridad.



Los certificados proporcionan un mecanismo para establecer una relación entre una clave pública y la entidad que posee la clave privada correspondiente. El formato más común de los certificados utilizados actualmente se define en la versión 3 del estándar internacional ITU-T X.509 (X.509v3). RFC 2459, un perfil de X.509v3, clarifica los campos definidos en X.509v3. La PKI de Windows 2000 utiliza el estándar X.509v3. Los certificados de Windows están programados de acuerdo a las aclaraciones especificadas en RFC 2459, pero aún así se les denomina certificados X.509v3. ITU-T X.509 no es la única forma de certificación. Por ejemplo, el correo electrónico seguro Pretty Good Privacy (PGP) se basa en una forma propia de certificados.

Entidad emisora de certificados. Certificate Authority (CA)

Una entidad emisora de certificados (CA) es una entidad a la que se le confía la emisión de certificados a un individuo, a un equipo o a otra entidad solicitante. Una entidad emisora de certificados acepta una solicitud de certificado, comprueba la información del solicitante de acuerdo con la directiva de la entidad y, a continuación, utiliza su clave privada para aplicar su firma digital al certificado. A continuación, la entidad emite el certificado al *subject* del mismo para que lo utilice como una credencial de seguridad dentro de una PKI. Puesto que las diferentes entidades emisoras de certificados utilizan métodos variados para comprobar el enlace entre la clave pública y el *subject*, es importante comprender las directivas de la entidad emisora de certificados (explicadas a continuación) antes de decidirse a confiar en dicha autoridad.

Una entidad emisora de certificados puede ser otra entidad como VeriSign. Por otra parte, puede ser una entidad emisora de certificados creada dentro de su organización, por ejemplo mediante la instalación de Servicios de Certificate Server de Windows 2000. Cada entidad emisora de certificados puede pedir distintos requisitos de prueba de identidad a los solicitantes de certificados. Por ejemplo: una cuenta de dominio de Windows 2000, una insignia de identificación de un empleado, una licencia de conductor, una solicitud con alguna certificación o una dirección física.

Directiva de entidad emisora de certificados (CA policy)

Para la emisión de certificados a los solicitantes, la entidad emisora sigue una serie de criterios establecidos. El conjunto de criterios que utiliza una entidad emisora de certificados a la hora de procesar las solicitudes de certificados (y de emitirlos, revocarlos y de publicar las CRL) se conoce como directiva de entidad emisora de certificados (CA policy). Generalmente, una entidad emisora de certificados publica su directiva en un documento conocido como Enunciado de práctica de certificación (Certification Practice Statement, CPS).

Tipos de (CAs) entidades emisoras de certificados

Los tres tipos de entidades emisoras de certificados son:

- **Entidad emisora de certificados con autofirma (self-signed CA)**. En una entidad emisora de certificados con autofirma, la clave pública del certificado y la clave utilizada para comprobar el certificado son la misma. Algunas entidades emisoras de certificados con autofirma son entidades emisoras raíz (root CAs)(consulte la tercera viñeta).
- **Entidad emisora de certificados subordinada (subordinate CA).** En una entidad emisora de certificados subordinada, la clave pública del certificado y la utilizada para comprobar los certificados son diferentes. Este proceso, donde una entidad emite un certificado a otra entidad emisora, se conoce como certificación cruzada (cross certification).
- **Entidad emisora de certificados raíz (root CA).** El cliente confía plenamente en una entidad emisora de certificados raíz, que ocupa la posición más alta en una jerarquía de certificados. Todas las cadenas de certificados terminan



en una entidad emisora de certificados raíz. La entidad raíz debe firmar su propio certificado porque no existe ninguna entidad emisora decertificados superior en la jerarquía de certificados. Todas las entidades emisoras de certificados con autofirma son entidades raíz, ya que la cadena de certificados termina cuando se alcanza una entidad emisora de certificados con autofirma.

Jerarquías de entidades raíz (rooted CA hierarchies)

Un administrador puede crear una jerarquía de entidades emisoras de certificados comenzando por un certificado de entidad emisora raíz y, a continuación, ir agregando entidades intermedias que emitirán certificados a entidades subordinadas. La cadena termina cuando una entidad emite un certificado a una entidad final (un usuario).

Los costos de distribución de los certificados de entidades raíz son los más elevados, ya que si se cambian las raíces hay que crear de nuevo toda la PKI. Si un certificado raíz cambia, debe revocar el antiguo certificado y agregar el nuevo para todos los clientes de la organización. Además, deberán volverse a emitir todos los certificados que las entidades raíz han emitido y que, a continuación, han pasado de las entidades subordinadas a las finales. Por tanto, el uso de un reducido número de certificados de larga duración emitidos por entidades raíz ofrece la solución más rentable a la hora de distribuir su jerarquía de entidades emisoras de certificados. Las entidades raíz son importantes porque se confía en ellas plenamente y porque son el punto donde se detiene la construcción de la cadena. Por tanto, es necesario que una autenticación fuera de banda acompañe a su distribución. Es decir, puesto que un certificado de entidad raíz está autofirmado, alguien debe garantizar la autenticidad de este certificado raíz.

Como existen muchas más entidades finales que entidades emisoras de certificados, las entidades que emiten certificados a las entidades finales utilizan su clave privada para firmar una gran cantidad de datos. Cuanto más se utilice una clave para firmar datos, más oportunidades hay de que se produzca un ataque criptográfico. Por tanto, el mantenimiento de la seguridad requiere que las entidades en línea que emitan certificados a entidades finales cambien las claves de firma con frecuencia.

Las entidades que emiten certificados a entidades finales tienen listas de certificados revocados mucho más largas que las entidades emisoras de certificados intermedias o raíz (que emiten certificados únicamente a otras entidades emisoras que se encuentran en una posición más subordinada). En parte esto se debe a que existen muchas más entidades finales que certificados de entidades emisoras. Además, existen muchas razones por las que las entidades finales deben revocar sus certificados, por ejemplo cuando un empleado cambia de trabajo o abandona la compañía.

Las entidades emisoras de certificados (CAs) publican una lista de certificados revocados (CRL), que es una lista de certificados que no deben usarse más. La entrada correspondiente al certificado permanece en la CRL hasta la fecha (Valid-To Date) especificada en el certificado y, a continuación, la entidad emisora de certificados la quita. Cuantas más entradas existan en una CRL, más larga será esta lista y más tiempo se tardará en efectuar la descarga. Generalmente, el tiempo de descarga sólo afecta a los usuarios con vínculos de red lentos, como los vínculos de acceso telefónico. La entidad emisora de certificados también puede administrar el tamaño de la CRL. Uno de los métodos consiste en mantener varias listas, conocidas como CRL divididas. El otro mecanismo consiste en reducir el período de validez de los certificados emitidos y, por tanto, acelerar su retirada de la CRL por parte de la entidad emisora de certificados.

Muchas de las aplicaciones deben poder averiguar la información de estado de revocación más reciente para un certificado. Únicamente una entidad emisora de certificados que esté en línea puede publicar la información actual acerca del estado de un certificado. El estado de revocación publicado desde una entidad emisora de certificados sin conexión debe distribuirse a una ubicación en línea mediante un método externo.

Las entidades emisoras más vulnerables son aquellas que se encuentran en línea, su seguridad física es baja y firman un gran número de certificados. Por tanto, cuando se establezcan las entidades emisoras de certificados raíz y las subordinadas, se debe lograr un equilibrio entre la seguridad y la disponibilidad. Lo más recomendable es establecer una jerarquía de tres niveles que incluya una entidad emisora de certificados raíz y una entidad de directiva subordinada, ambas independientes y sin conexión, y otra entidad de empresa en línea que emita certificados a la subordinada.

Entidades emisoras raíz sin conexión (offline root CA). Diseñe la jerarquía de entidades emisoras de certificados otorgando a las entidades raíz la seguridad más elevada. Mantenga una entidad emisora raíz sin conexión en un lugar seguro y utilícela para firmar únicamente un número reducido de certificados. Preferiblemente, mantenga la CA y las claves en un lugar cerrado al cual deban tener acceso al menos dos operadores: uno para llevar a cabo las acciones necesarias y otro para auditarlas. (En una red de Windows 2000, las entidades independientes están diseñadas para funcionar sin conexión.)

Entidades emisoras intermedias sin conexión (Offline intermediate CAs). Debajo de la entidad raíz puede haber una o más entidades subordinadas. Al convertir las entidades emisoras de certificados subordinadas intermedias en máquinas sin conexión, aumentará la seguridad de estas entidades.

Entidades emisoras en línea (Online issuing CA). La última entidad emisora de certificados de la cadena debe estar en línea y, por tanto, disponible para procesar nuevas solicitudes de certificados procedentes de diversos clientes. Una entidad emisora de certificados en línea, situada debajo de la entidad raíz, también puede publicar con frecuencia información actualizada acerca del estado de revocación. El administrador puede cambiar a menudo las claves para este tipo de entidades emisoras de certificados puesto que el costo (en cuanto a carga administrativa) de distribución de sus nuevos certificados es mínimo. Las entidades emisoras de certificados subordinadas no son realmente desechables, pero pocas son las ventajas que se obtienen atacándolas, y pueden bloquearse rápida y fácilmente si la revoca una entidad superior.

Registración

Registrarse es el proceso por el cual los *"subjects"* se dan a conocer a una entidad emisora de certificados (CA). El registro puede estar implícito en el acto de solicitud de un certificado, puede llevarse a cabo mediante otra entidad de confianza (como una estación de inscripción de tarjetas inteligentes) que avale el *subject* o bien puede llevarse a cabo como información resultante recibida de una fuente de confianza (como un administrador de dominio). Una vez registrado con la entidad emisora, se emite un certificado al *subjects* siempre y cuando el certificado cumpla con los criterios establecidos en la directiva de entidades emisoras de certificados.

Inscripción de certificados (Certificate Enrollment)

La inscripción de certificados es el procedimiento por el cual una entidad final solicita y recibe un certificado de una entidad emisora de certificados. La solicitud de certificados proporciona información de identidad a la entidad emisora de certificados. A continuación, esta información pasa a formar parte del certificado emitido. La entidad emisora de certificados procesa la solicitud conforme a una serie de criterios que pueden requerir que la autenticación no automática tenga lugar sin conexión (autenticación fuera de banda). Si la solicitud se procesa correctamente, la entidad emite el certificado al usuario.

Inscripción de certificados a través de las autoridades de registro

Con una Autoridad de registro (Registration Authority, RA) se realiza un tipo especial de inscripción de certificados que garantiza a una entidad emisora de certificados el enlace entre las claves públicas y la identidad y los atributos del posible propietario de un certificado. Las autoridades de registro son *subjects* que poseen un certificado especial. Este certificado especial contiene información que identifica el *subject* ante la entidad emisora de certificados. Generalmente, una autoridad de registro establece el enlace entre la clave pública y el *subject* y, a continuación, firma la solicitud generada por el *subject* para probar a la entidad emisora de certificados que la autoridad de registro está garantizando el enlace. En esencia, el uso de la autoridad de registro es una forma de delegación administrativa ya que la entidad emisora de certificados delega a la autoridad de registro la comprobación del enlace entre una clave pública y una entidad.



La autoridad de registro genera certificados que establecen un fuerte enlace entre el tema y la clave pública. En el sistema operativo Windows 2000, Servicios de Certificate Server utiliza la autenticación de dominio para identificar al usuario que efectúa solicitudes para la mayoría de los tipos de certificados. Un autoridad de registro utiliza la estación de inscripción de tarjetas inteligentes de Windows 2000, que crea certificados de inicio de sesión con tarjetas inteligentes. La autoridad de registro valida la identidad del destinatario de la tarjeta inteligente, lo que

proporciona un enlace más fuerte entre el *subject y la clave pública que la que se puede conseguir si sólo se utiliza autenticación de dominio.*

Revocación de certificados

Los certificados tienen una duración específica, pero las entidades emisoras de certificados pueden reducir esta duración mediante el proceso conocido como revocación de certificados. La entidad emisora de certificados publica una lista de revocación de certificados (CRL) con los números de serie de aquellos certificados que considera que ya no pueden utilizarse más. La duración establecida para las listas de revocación de certificados suele ser mucho más corta que la que se establece para los certificados. La entidad emisora de certificados también puede incluir en la CRL la razón por la que se revocó el certificado. También incluye una fecha a partir de la cual se aplica este cambio de estado.

Para revocar un certificado se pueden indicar las razones siguientes:

Riesgos para la clave

Riesgos para la entidad emisora de certificados

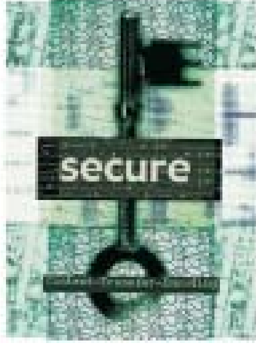
Cambio de afiliación

Reemplazo

Cese del funcionamiento

Conservación de certificados (éste es el único código que le permite cambiar el estado de un certificado revocado; resulta útil si el estado del certificado es cuestionable)

La revocación de un certificado por parte de la entidad emisora de certificados significa que dicha entidad da de baja su instrucción acerca del uso permitido del par de claves antes de que el certificado caduque normalmente. Una vez que el certificado caduca, se quita su entrada de la CRL para reducir el tamaño de esta última.



Durante la comprobación de la firma, las aplicaciones pueden comprobar la CRL para determinar si un par de claves y un certificado dados son todavía de confianza (algunas aplicaciones usan la API de comprobación en cadena de Microsoft incluida en CryptoAPI para hacerlo). De no ser así, la aplicación puede determinar si la razón o la fecha de la revocación afecta o no al uso del certificado en cuestión. Si el certificado va a utilizarse para comprobar una firma y la fecha de la firma figura es anterior a la fecha de revocación del certificado por parte de la entidad emisora de certificados, la firma todavía se puede considerar válida.

Después de que una aplicación obtiene una CRL, el cliente almacena en memoria caché dicha lista y la utiliza hasta que caduca. Si una entidad emisora de certificados publica una nueva CRL, las aplicaciones que tienen una CRL válida no usan la nueva CRL hasta que caduca la que ya tienen.

PKI en Windows 2000-2003 Server.

Muy probablemente UD sabrá o habrá escuchado que Microsoft ha incluido componentes de la infraestructura PKI en sus sistemas operativos desde Windows 2000. La pregunta es que puede PKI hacer por Ud y en que y como lo utilizo con Windows 2003.
recordemos que

1. PKI es una manera de autenticar basada en Estándares e independiente de cualquier sistema operativo

2. Aunque tiene gran potencialidad, PKI no es usado extensivamente en los sistemas operativos de Windows Server, salvo en algunas pocas aplicaciones.

Para contestar "que puede hacer PKI para uno"? entendamos que la idea de un "certificado" es la de un tipo de tarjeta de identificación. Un pasaporte que prueba que uno es quien dice ser con un cierto grado certeza. Pero no solo sirven para identificar personas. Sirven para identificar servidores y cualquier otra cosa que necesite probar su identidad.

La idea de "cerificados" no es una idea de Microsoft. Las ideas sobre PKI han sido desarrolladas desde hace muchos años en el mundo Unix. Esto es muy bueno ya que es una infraestructura ya muy bien probada y Microsoft se ha basado en las últimas versiones de tecnologías ya muy bien comprendidas. Por ejemplo el formato de los certificados tomado por Microsoft es X.509.

PKI en Windows 2003 Server es aún algo en progreso. Si uno quisiese PKI-izar todo en Windows 2000-2003 sería imposible. Uno por ejemplo no puede detener que los Domain Controlers se autentiquen vía Kerberos (ver articulo en NEX 7 sobre autenticación Kerberos) y comenzar a utilizar PKI. Si podría setear a sus usuarios para utilizar certificados usando "smart cards". A continuación describimos qué cosas son posible utilizar bajo Windows 2003 que usen PKI:

Crear y usar certificados para permitir que dos sistemas de diferentes dominios usen IPSec (leer un excelente articulo de IPSec en NEX1). Como sabemos IPSec es una excelente manera de autenticar y/o encriptar comunicaciones IP entre dos sistemas (dos o más) a través de Internet. Pero esos sistemas necesitan ser capaces de autenticarse y Microsoft ofrece tres opciones: ambos lados comparten una clave secreta (un password), ambos se autentican vía Kerberos (no esta mal pero esto funciona solo si los sistemas pertenecen al mismo forest (bosque). O vía certificados. Usar certificados es la única manera en que dos sistemas que no son miembros del mismo bosque pueden hacerlo vía IPSec.

Asegurar el acceso Web. Una de las maneras en que uno puede controlar el acceso a los sitios Web es vía certificados.

Crear y usar certificados para asegurar el uso de e-mail dentro de una organización. Uno puede crear certificados para usuarios de e-mail que estos pueden usar bien para autenticarse o encriptar las comunicación vía e-mail. El único problema es que ambos interlocutores deberán aceptar como validos los certificados creados por "alguien". Por supuesto esto es posible si hablamos de una o dos compañías que son socias. Pero no funcionaria para asegurar el e-mail al mundo. Para ello deberemos "confiar" (to trust) en una fuente única de certificados otorgados por empresas como Thawte.

Logons via smart cards: Sirven entonces los certificados para logonearse? Esta es de algún modo la autenticación mas común que realiza alguien para acceder a su cuenta. Por supuesto que es posible pero como entramos el "certificado"? En un logon "clásico" uno tipea cuenta (nex@nexweb.com.ar o nex) y un password asociado. Entrar además el certificado puede ser muy complejo ya que involucra miles de bytes. Entonces uno podrá logonearse a una cuenta en un dominio con un certificado solo si éste está almacenado en algo similar a una tarjeta de crédito llamado "smart card". Además, necesitamos que la maquina donde nos logoneamos tenga una lectora de "smart cards". Y eso cuesta.

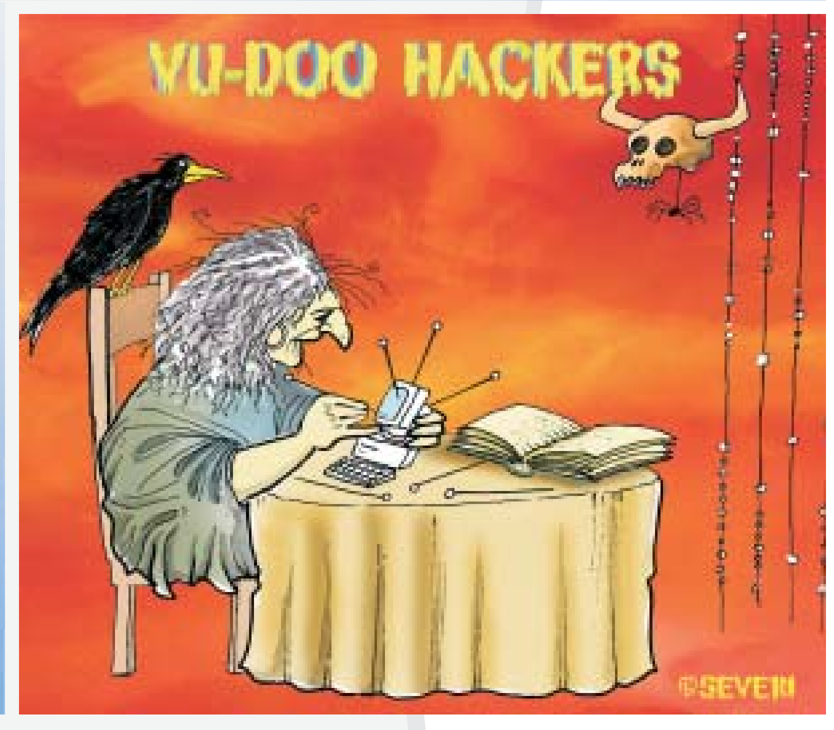
Agentes de recuperacion de EFS (encripted File system): El EFS de Microsoft nos permite elegir personas que pueden desencriptar nuestros archivos en el caso que uno no recuerde el password por ejemplo. Uno designa nuevos Recovery Agents (agentes de recuperación) vía el uso de certificados y PKI.

Firmar programas: Como sabrá desde Windows 2000 es posible firmar un dado programa, un programa de instalación de software o un driver. De este modo me aseguro el origen del programa o si ha sido testeado por los laboratorios de Microsoft. El firmado se hace vía certificados.

Como ya dijimos todo esto está en evolución. Y muy probablemente en un futuro cercano podremos llevar una smart card con un certificado personal. Supongamos que tenemos una cuenta en una red Novell, una en un forest Active Directory de Microsoft y una cuenta de Amazon.com. PKI funcionaría de este modo: entregaríamos una copia de nuestro certificado al administrador de Novell otra al de AD, al de Amazon y cualquier otro donde tengamos una cuenta de usuario. Ellos harán una asociación entre ese certificado y la cuenta en su infraestructura (algo que se llama mapear un certificado a una cuenta). Desde ahí tendríamos un solo password de que preocuparnos!!!!

Validación de la cadena de certificados en Windows 2000-2003

Microsoft CryptoAPI ofrece un marco estándar que las aplicaciones pueden usar para obtener servicios criptográficos y de certificación digital. Además de los servicios estándar que CryptoAPI proporciona para Windows 95, Windows 98, Windows NT y Windows 2000, otros fabricantes pueden desarrollar y comercializar sus propios módulos complementarios para proporcionar servicios criptográficos adicionales. En una red de Windows, cuando se genera una solicitud para un certificado nuevo, la información de ésta se pasa primero del programa de solicitud a CryptoAPI. A continuación, CryptoAPI pasa los datos adecuados a un programa conocido como proveedor de servicios criptográficos (CSP) instalado en el equipo o en un dispositivo, como una tarjeta inteligente, al que pueda tener acceso el equipo. Un CSP es un módulo de software independiente que realiza operaciones de cifrado, como el intercambio de claves secretas, la firma digital de datos y la autenticación de claves públicas. Cualquier servicio o aplicación de Windows 2000 puede solicitar las operaciones de cifrado de un CSP.

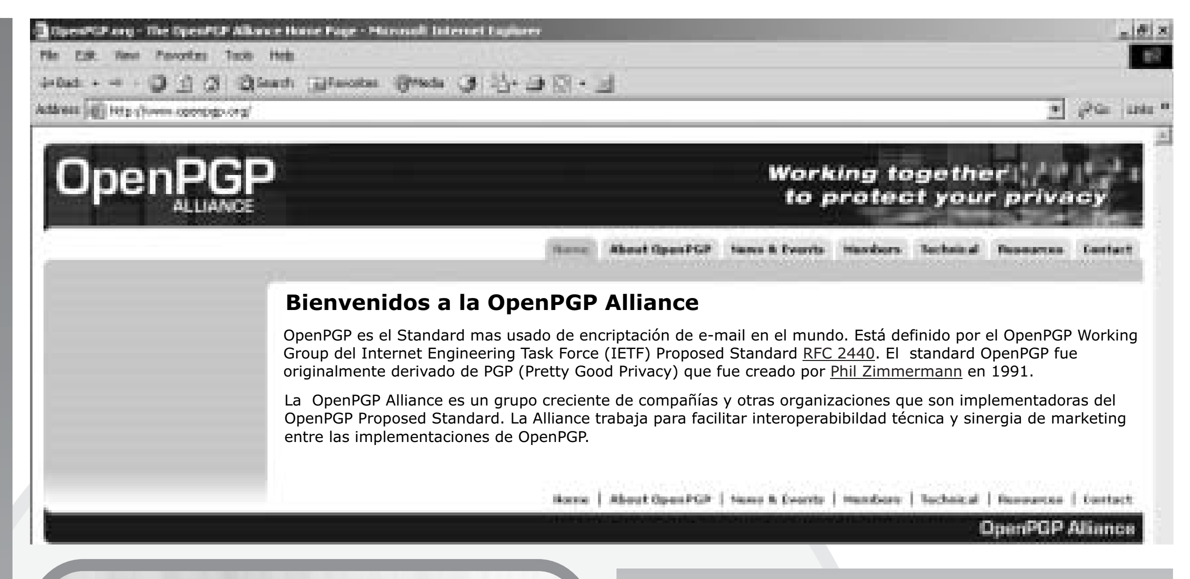


PKI en Windows 2000-2003 Server (cont.)

El mecanismo de generación de cadenas de certificados de Windows 2000 intenta generar una ruta de certificación (una cadena de certificados) desde el certificado de la entidad final, como un certificado de usuario, hasta un certificado raíz de una entidad emisora de certificados, de la siguiente manera:

1. El mecanismo de encadenamiento de certificados consulta el campo Authority Key Identifier (AKI, Identificador de clave de la entidad emisora) del certificado (si existe) para buscar el certificado primario (emisor) en uno de los almacenes de certificados del sistema. Si el AKI contiene valores para los campos Issuer (Emisor) y SerialNumber (Número de serie), el mecanismo de encadenamiento de certificados busca un certificado primario específico en los almacenes del sistema; de lo contrario, usa el valor del identificador de clave para buscar un certificado primario coincidente. Si no existen valores en el certificado para los campos Issuer (Emisor), Serial Number (Número de serie) y Key Identifier (Identificador de clave), el mecanismo de encadenamiento de certificados usa el campo IssuerName (Nombre del emisor) para buscar un certificado primario coincidente.
2. Si el mecanismo de encadenamiento de certificados no encuentra un certificado primario en los almacenes del sistema basándose en los campos descritos anteriormente, consulta el campo Authority Information Access (AIA, Acceso a la información de entidad emisora) del certificado. Utiliza la información que encuentre allí (si la hubiera) para recuperar el certificado primario de la ubicación especificada, como una ubicación de Protocolo de transferencia de hipertexto (HTTP) o una ubicación de Protocolo compacto de acceso a directorios (LDAP).

Este proceso se repite para cada certificado primario que se encuentre encima del certificado de entidad final del usuario y finaliza con un certificado con autofirma que reside en el almacén raíz local de confianza. Un certificado con autofirma es un certificado cuyo nombre de tema y de emisor coinciden, y cuya clave pública puede utilizarse para comprobar la firma que se incluye con el certificado.



PGP, Pretty Good Privacy

Que es PGP?

Pretty Good Privacy (PGP) es un paquete de software desarrollado por R. Zimmermann que provee rutinas cryptograficas para e-mail y aplicaciones de almacenamiento de archivos. Lo que hizo Zimmerman es tomar cryptosistemas ya existentes y protocolos criptográficos y desarrolló un programa que puede correr en múltiples plataformas. Provee encriptación de mensajes, firmas digitales, compresión de datos y compatibilidad de e-mail.

Los algoritmos que utiliza por default (especificados en el RFC 2440) son : ElGamal y RSA para el transporte de llaves y triple-DES, IDEA y CAST5 para la encriptación de mensajes. Las firmas digitales se consiguen utilizando DSA para firmar y SHA-1 o MD5 para la computación de los hashes de los mensajes. El programa shareware ZIP es utilizado para comprimir mensajes para transmitirlos y almacenarlos. La compatibilidad con E-mail se logra con el uso de la conversión Radix-64.

Las versiones en US de PGP han sido limitadas por leyes Federales de exportación debido al uso de crysto sistemas controlados para la exportación. Pero, recientes relajaciones han permitido eliminar varios de los obstáculos.

Philip R. Zimmermann creador de PGP

Philip R. Zimmermann es el creador de Pretty Good Privacy (PGP) (Privacidad Bastante Buena). Por haber hecho esto, fue blanco de una investigación criminal durante tres años. Esto porque el gobierno americano sostenía que las restricciones para exportación de software de criptografía había sido violado cuando PGP se distribuyó por todo el mundo seguido a su publicación como freeware en 1991. A pesar de su falta de recursos, falta de personal pago, falta de una empresa para respaldarlo y persecución gubernamental, PGP así y todo se transformo en el software para encriptación de e-mails mas popular del mundo. Luego que el gobierno retiro los cargos en 1996, Zimmeran fundó PGP Inc. La empresa fue luego adquirida por Network Associates Inc. (NAI) en Diciembre 1997. El permaneció como un Senior Fellow por tres años más. En 2002 PGP fue adquirida por una nueva empresa llamada PGP Corporation, donde Zimmerman ahora realiza tareas de consulta. En la actualidad realiza independientemente tareas de consultaría para una serie de empresas y organizaciones industriales en asuntos de criptografía. Es actualmente un Fellow de Stanford Law School's Center for Internet and Society.

Antes de fundar PGP Inc, Zimmerman era un ingeniero de software con más de 20 años de experiencia, especializándose en criptografía y seguridad de datos, comunicación de datos y sistemas embebidos en tiempo real. Su interés en el lado político de la criptografía, nació de su pasado en asuntos relacionados con políticas militares.

Ha recibido numerosas premios tanto técnicos como humanitarios por su trabajo pionero en criptografía.

Zimmermann recibió su titulo de Licenciado en Ciencias de la computación de la Florida Altantic University en 1978. Es miembro de numerosas organizaciones: International Association of Cryptologic Research, la Association for Computing Machinery y la League for Programming Freedom. Es actualmente el chairman de OpenPGP Alliance y sirve en el Board of Directors for Computer Professionals for Social Responsibility, y en el Advisory Boards for Anonymizer.com, Hush Communications, Veridis y Qualys. (mas info en http://www.philzimmermann.com)

Porque el PKI de OpenPGP es mejor que el PKI de X.509

Philip Zimmermann
27 Feb 2001

En la mente de mucha gente, la frase "Public Key Infrastructure" se ha convertido en sinónimo de "Certificate Authority" (CA). Esto es porque en el mundo X.509, el único PKI con que nos encontramos esta construido alrededor del CA. Matt Blaze hizo la siguiente observación: "los CA comerciales nos protegerán de cualquiera al que ese CA se niegue a aceptarle dinero". Estos CAs están "incluidos dentro" de la mayoría de los browsers, sin que el usuario pueda decidir de confiar en ellos o no.

A lo largo de este articulo, nos referiremos a OpenPGP bajo el standard IETF en lugar de PGP, que es una implementación particular del Standard OpenPGP.

Existe, una Public Key Infrastructure OpenPGP. Pero lo que llamamos una PKI en el mundo openPGP es en realidad la amalgama que surge de la suma total de todas las llaves en la población de usuarios, todas las firmas en todas esas llaves, las opiniones individuales de cada usuario de OpenPGP sobre a quien eligen como "introducers" confiables ("trusted introducers"), todas los softwares clientes que corren el modelo de confianzas openPGP y realizan cálculos sobre confianzas para cada usuario cliente y los servidores de llaves que en forma fluida diseminan este conocimiento colectivo.

PGP ha crecido por muchos años sin la necesidad de establecer un CA centralizado. Esto es porque OpenPGP usa un sistema de "trusted introducers", que son equivalentes a un CA. OpenPGP permite a cualquiera firmar la llave pública de cualquier otro. Cuando Alice firma la llave de Bob, ella esta "introduciendo" la llave de Bob a cualquiera que confía en Alicia. Si alguien confía en Alicia para introducir llaves, entones Alice es una "trusted introducer" en la mente de ese observador.

Si yo obtengo una llave que ha sido firmada por varios "introducers" y uno de ellos es Alice, y yo confio en Alice, entonces esa llave esta certificada por un "trusted introducer". Puede estar firmada por otros "trusted introducers", pero yo no confio en ellos, de modo que no son "trusted introducers" desde mi punto de vista. Es suficiente que Alice haya firmado la llave ya que yo confio en Alice.

Seria aun mejor si dentro de los varios "introducers" de esa llave se incluyeran dos o mas personas que yo confiara. Si la llave esta firmada por dos "trusted introducers", entonces estaré más confiado en la certificación de esa llave, ya que es mas improbable que una atacante pudiera engañar a dos "introducers" de mi confianza a firmar una llave "trucha". Las personas pueden ocasionalmente cometer errores y firmar la llave errada. OpenPGP tiene una arquitectura "fault tolerant" (a prueba de fallo) que me permite exigir que una llave este firmada por dos "trusted introducers" para que sea considerada valida. Esto permite un grado aun mayor de confianza en que la llave pertenece a la persona nombrada en la llave.

Por supuesto, un atacante inteligente podría engañar a dos o mas "introducers" no muy sofisticados a firmar una llave publica "trucha". Pero, eso no es importante en el modelo de confianzas de OpenPGP, ya que yo no confio en "introducers no sofisticados que pueden ser engañados muy fácilmente. Nadie debiera. Uno solo debe confiar en "introducers" honestos y sofisticados que entienden lo que significa firmar una llave, y ejercitaran seriedad en verificar la identidad del poseedor de la llave antes de firmar la llave en cuestión.

Si solo "introducers" no confiables firman llaves "truchas" nadie será engañado en el modelo de confianzas de PGP. Uno debe decirle al software OPenPGP cliente que "introducers" son de confianza.

El software cliente usara ese conocimiento para calcular si una llave esta certificada propiamente por un "introducer" confiable mirando las firmas de uno de los "introducers" de confianza. Si la llave no posee firmas de "introducers" que uno le ha dicho al software que confía, el software cliente no considera la llave como certificada y no lo dejara usarla (o por lo menos le indicara a no usarla).

Cada uno elige a quien considera un "introducer" de confianza.

En muchos casos habrá solapamiento, ya que muchos "introducers" se transforman en confiables para un amplio espectro. Podrían hasta firmar un gran número de llaves como una ocupación full-time. Esos son llamados CAs en el mundo X.509.

No hay nada malo con tener CAs en OpenPGP. Si mucha gente elige confiar al mismo CA para que actúe como un "introducer", y ellos todos configuran sus propias copias del software cliente de OpenPGP para confiar a ese CA, entonces el modelo de confianzas OPenPGP actúa en idéntica manera que el modelo X.509. De hecho, el modelo de confianza OPenPGP es un "superset" (inversa de subconjunto) del modelo de confianzas centralizado que normalmente vemos en el mundo X.509. No existe ninguna situación en el modelo de confianza X.509 que no puede ser tratado de idéntico modo en el model de confianza de OPenPGP. Pero, OpenPGP puede hacer mucho mas, y con una arquitectura "fault tolerant", y mas control del usuario sobre su perpectiva del model PKI.

E-MAIL SERVERS EN LINUX

El e-mail (correo electrónico) se ha transformado en una herramienta casi indispensable debido en gran parte a su velocidad, es gratis y simple de usar.

Hay 2 componentes que deben funcionar para "hacer e-mail". Por un lado el "cliente" (software que utilizan quienes envían y reciben correo). Ejemplos de "clientes": Outlook Express, Outlook de Microsoft ó Eudora y una variedad de clientes del mundo Open-Source. Para Linux: Evolution, Pine (desarrollado por la Washington University) son sólo algunos ejemplos. Por el otro existe una infraestructura de servidores que realizan el envío, recepción y distribución. Ejemplo es Exchange 2000 de MS. Nuevamente el mundo Linux tiene varias alternativas. Algunas comerciales y otras Open Source y gratuitas. En la tabla 1 se puede ver una lista (en las diferentes columnas) de las mas reconocidas. Cada fila nos da su precio junto a las características más importantes a tener en cuenta. La última fila nos da una clasificación de 1 a 10 (10 siendo considerado el mejor).
Cuando se envía un e-mail este viaja entre 2 servidores de mail (mail servers). Se utiliza un protocolo conocido como SMTP (Simple Mail Transfer Protocol) que como lo indica su nombre realiza la transferencia entre los servidores. Como se distribuye el mail una vez que llegó a destino dependerá de cómo fue configurado.

Funciones de los mail servers

Los servidores de correo realizan dos funciones diferentes
- Relay (de posta)
- Collection Server (Recolección)

Lo normal es que un servidor cumpla ambos roles (de "relay" y "collection"). Uno puede "levantar" un servidor **SMTP** propio para enviar y recibir correo pero se debe poner extremo cuidado en su configuración si no se quiere perder mails o enviar mails incorrectamente. Es muy importante al configurar el "relay" de no permitir que cualquiera (*everybody*) pueda acceder a él y usarlo como *relay*. Los que realizan *Spam* muy probablemente lo usarán y será "ese" server el culpable de la acción.

Relay
Un ¨Relay Mail Server¨ hará un ¨forward¨ de un e-mail a otro servidor SMTP. Esto Sucede cuando los clientes se conectan a un SMTP server para enviar correo. La mayoría de los ISPs (Internet Service Providers, Proveedores de Internet) tienen un SMTP ¨Relay¨ server. Podría ser que un usuario tuviese su propio SMTP server y NO necesitase usar un ¨Relay¨.

Collection Server
Cuando llega un *e-mail* (por ejemplo a *jose@cortech.com.ar*) a un collection-server que regentea un dominio dado (p.ej: *cortech.com.ar*), éste lo distribuye al mailbox (buzón) correspondiente, esperando que el usuario José lo recoja usando el protocolo **POP3** (Post Office Protocol3), **IMAP4** (Internet Message Ardes Protocol), o un cliente de mail basado en comandos de `shell` como Mutt.

Spam y Virus

Un mail server debe poder ayudar a combatir:
- Spam
- Virus
Spam son e-mails que le llegan al usuario y que él no ha solicitado. Spam es un término muy usado para el llamado email no solicitado (**U**nsolicited **B**ulk o **C**omercial): **UBE** y **UCE**.
Es casi imposible que el servidor de correo filtre 100% el Spam pero deberá sí limitarlo lo máximo posible.
Un modo de evitar Spam es usar listas llamadas Realtime Blackhole List (**RBL**). Un ejemplo es **MAPS** que le permite al servidor de mails hacer **DNS** lookups de **ISP** y ver si están en las listas o no. Este es un servicio pago aunque existen alternativas gratis como **DNSRBL**.
Los virus son una amenaza que llega junto a nuestros e-mails. Una vez que entran a nuestro sistema pueden atacarlo y/o usarlo como base para atacar otros sistemas. El mail server puede ayudar intentando verificar si un virus viene en el mail. Existen muchos programas comerciales que permiten chequear por virus los e-mails que le llegan. A continuación analizaremos los servidores de correo más populares. Algunos son Open Source y otros comerciales. Haremos una breve descripción y analizaremos sus características referidas a: seguridad y capacidad de filtrado (Spam y virus)

Servidores de e-mail conocidos

QMail
Un servidor de correo que fue diseñado bajo las normas impuestas en los RFC de la IETF.
URL: www.qmail.org
Costo: $0,-



QMail fue diseñado respondiendo en un 100% a los RFC (Request For Comments) de la IETF. Se destaca sus características de seguridad al punto que existe una recompensa de U$10,000 para cualquiera que pueda realizar un exploit de qmail. Aun esa recompensa permanece sin haber sido reclamada.
Instalar `qmail` es simple aunque no es el estándar "`./configure && make`". Configurarlo resulta aún más fácil. Tiene una opción en el `make` de modo de poder colocar algunas entradas de configuración. Con esto es suficiente para arrancarlo funcionando en una configuración básica. **QMail** tiene archivos de configuración, mayormente de una línea en `/var/qmail/control`. Allí entramos dominios y hostnames para envío y recepción. Cada archivo tiene un nombre simple, por eso es fácil adivinar cuál se necesita editar para un propósito específico.
QMail puede manejar reparto por `mbox` o `Maildir`. Puede además canalizar su mail a través de cualquier aplicación que usted quiera, como por ejemplo `SpamAssassin`.
Aunque no tiene ningún soporte para **DNSRBL** o para chequear los virus, es fácil configurar `qmail` para que mande todos los mails vía un chequeador de virus o un filtro de correo electrónico "basura" (*spam filter*).
Uno de los mayores problemas con qmail es que levanta (*forks*) un proceso para cada mail individual que trate de enviar. Ya sean sea locales o remotos. Por defecto, simultáneamente, `qmail` enviará 20 mensajes remotos y 10 mensajes locales, lo cual debería ser más que suficiente para la mayoría de los sistemas. `qmail` hace básicamente reparto de mails pero existen más que suficientes parches disponibles ofreciendo encriptación **TLS**, configuración *MySQL* y conectividad **IPv6** como también mejoras en el rendimiento y almacenamiento de colas para sistemas con grandes cantidades de mail.
Si utiliza los parches de *http://kldp.org/-enunjea/qmail_cocktail.php*, `qmail` es un servidor de mail capaz de poder manejar grandes volúmenes de trafico. Muchas grandes organizaciones usan `qmail`.

SendMail
Es el servidor de mail más popular de todo el mundo UNIX.
URL: www.sendmail.org
Costo: $0



Sendmail es el nombre de servidor de correo más popular del mundo Linux (Unix en general). Y, se instala por defecto en casi "todas" las distribuciones.
Es conveniente saber manejar sendmail y quizas luego usar otro. Escalea muy bien y tiene un sinnumero de características, incluyendo el soporte **DNSRBL**.
El mayor problema con sendmail es que su sistema de configuración está basado en ´`m4'`. Este es un leguaje macro no difícil de dominar. Pero si implica que usted debe dedicarle una significativa cantidad de tiempo para descifrar cómo configurar sendmail de una forma segura y que haga lo que usted desee. A menos que uno desee tener un `sendmail` muy a medida tener que programar en `m4` no es una buena opción.
Por suerte la configuración por defecto es suficientemente buena para tener un servidor de correo básico.
Si se puede dominar `m4` (a través de la documentación `sendmail` o uno de los numerosos libros que existen) `sendmail` es un servidor muy poderoso y capaz. Como es tan popular existe una comunidad de usuarios de `sendmail` en internet muy grande y conocedora capaz de ayudarnos.
Muchas distribuciones se están alejando de `sendmail` por sistemas más simples como **Postfix** y **Exim**.
Si usted elige `sendmail`, asegúrese de estar al tanto con lo último en seguridad, ya que `sendmail` no tiene gran fama con los *exploits*. De todos modos si uno actualiza constantemente los sucesivos parches nos cubren.

Postfix
Es muy fácil de configurar y está resultando un reemplazo para sendmail.
URL: www.postfix.org
Costo: $0



Postfix comenzó originalmente como una alternativa (de más simple configuración) a `sendmail`. Desde su concepción como **VMailer** se ha desarrollado en un servidor de mail muy bueno y de sencilla puesta en marcha.
Cambiar a Postix si esta sendmail funcionando es relativamente directo aunque deberemos reconfigurar bastante.
Postfix puede ser usado para un amplio rango de aplicaciones, desde un simple servicio de mail, hasta un servidor de mail en gran escala. Lo más común es verlo como un simple re-enviador en workstations.
Postfix puede fácilmente estar configurado para que lea opciones específicas de configuración, incluyendo dominios virtuales y aliases desde un archivo externo a la configuración principal. De esta forma es fácil agregar más dominios.
Desafortunadamente, más allá de `dbm`, Postfix no soporta el uso de bases de datos como *MySQL* o *PostgreSQL*, para almacenar detalles de configuración, por esto no es ideal si se desea realizar una reconfiguración real-time.
Mandrake Linux ha usado ***Postfix*** durante bastante tiempo como servidor **SMTP** por defecto.Esta facilidad en la configuración ha hecho maravillas en su uso, y por esto,



la mayoría de las personas ya no lo abandonarán. Para configuración simple de mail servers Postfix es una opcion excelente. Si lo que uno está manejando es un gran número de dominios y quiere configuraciones complejas de dominios virtuales otros servidores pueden ser más apropiados.

Exim
Es muy poderoso, escalable y bien documentado.
URL: www.exim.org
Costo: $0



Fue desarrollado en la Universidad de Cambridge, UK. ***Exim*** es un servidor de mail muy poderoso que ha sido diseñado para manejar grandes cantidades de mail y para limitar a **UCE**/**UBE** (*spam*) en la red. Muchas empresas que tienen gran cantidad de tráfico, incluyendo *Sourceforge*, usan ***Exim***. Esto se debe básicamente por como escalea y ya que puede ser ajustado específicamente para las necesidades de los usuarios. Aunque usted tenga cinco usuarios detrás de su firewall o diez mil cuentas, Exim puede manejar un gran rango de aplicaciones y ciertamente puede lidiar con casi todo lo que le den.

Exim tiene una estructura de configuración muy completa, ofreciendo de todo desde ACLs (*Access Control Lists*) a soporte de base de datos desde *MySQL* y *PostgreSQL* a *Oracle* y *DB2* de *IBM*.

Sin embargo, no hay necesidad de aprender todos los aspectos de configuración para tener a Exim funcionando, ya que hay gran cantidad de documentos disponibles en el sitio de ***Exim***. Expandir las capacidades del servidor es realmente fácil y la flexible estructura de configuración da una enorme cantidad de opciones para cada segmento de la organización.

Muchas distribuciones, incluyendo *Debian* ofrecen *Exim* y sus configuraciones por defecto son muy completas y seguras. Estas distribuciones ofrecen herramientas de configuración que simplifican enormemente el setup. Como con la mayoría de las cosas del mundo *Linux*, *O´Reilly* ha producido un libro dedicado a *Exim*. Si Ud instala *exim* es un "*must*". Aunque, la documentación propia del sitio de *Exim* es para recomendar.

Exim es uno de los más poderosos sistemas de mail *Open Source* disponibles. Si se lo compara con **`sendmail`** debemos destacar la sencillez de su configuración. Y, no hay ninguna característica esencial que no tenga.

Esto lo diferencia de ***qmail*** ya que uno debe en este último depender de esfuerzo de terceros para tener ciertas caracteristicas especiales.

SuSE Linux eMail Server 3.1
Toda su configuración puede hacer vía una interfase web.
URL: www.suse.com
Costo: U$S 999,-



SuSE siempre ha sido bien conocido por su muy popular distribución de Linux. Por esto no es sorprendente que ofrezcan un producto excelente para servidor de correo.

Como muchos de sus otros productos, están apuntando con este servidor a compañías de tamaño medio que puedan mantener su propio sistema de email y pagar los fees correspondientes.

El sistema ofrecido por *SuSE* está basado en un servidor **SMTP** e **IMAP**. Esto se complementa con una muy simple administración mediante un Web *front-end*. Todo lo que el administrador necesita hacer puede hacerse por medio de esta interfase.

Por tanto no hay necesidad de configuración manual o investigaciones a fondo de las complejidades de SMTP. Por supuesto, entender es siempre requerido y, ciertamente uno no podría utilizarlo sin un poco de conocimiento sobre cómo funciona email. Pero no se requiere un nerd de redes o de Linux para hacer que funcione con éxito.

Junto con los emails, la interfase maneja un calendario, la agenda de direcciones y una lista de tareas a realizar.

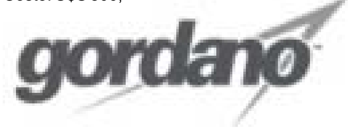
A pesar de que cada una de estas son características son posible realizarlas por otros productos el hecho es que tenerlo disponible en un solo lugar, accesible a través de la red significa una ventaja para cualquiera que administre un servidor de mail.

SuSE especifica que su sistema puede manejar alrededor de dos millones de usuarios, con entre cincuenta y doscientas personas accediendo al sistema simultáneamente montando un servidor de capacidad razonable. Esto es más que suficiente en la mayoría de las situaciones.

Este sistema está actualmente basado alrededor de Postfix utilizando un servidor LDAP para uso de la administración de usuarios. Por esto no es posible construir un sistema similar Open Source sin tener que gastar dinero en desarrollo.

Como uno esperaría, es la facilidad de la configuración la que atrae a las personas a este sistema de SuSE. Aunque por su costo de U$S 999,- vale la pena ver las alternativas de *Open Source*.

GLMail
Poderoso, fácil de usar y de instalar
URL: www.ntmail.co.uk
Costo: U$S 800,-



Gordano ha producido durante varios años un servidor de mail muy popular para Windows NT. Recientemente ha decidido migrarlo a Linux y otras plataformas Unix.

La lista de características para GLMail es impresionante junto a su facilidad de instalación y configuración.Es ideal para aquel que deasea un mail server en servicio, pero no desea perder el tiempo tratando de comprender qué está haciendo.

La instalación es un proceso simple y GLMail no depende de ninguna distribución específica de Linux.

Esto lo hace muy interesante para uno que está familiarizado con un estilo particular de Linux y no desea cambiar para solo instalar su servidor de mail.

Todo es manejado a través de una interfase Web, permitiendo el control completo del servidor. Esto mantiene escondida del administrador los detalles de configuración.

GLMail es particularmente popular por su capacidad de filtrado anti-virus y anti-spam. Es bueno bloqueando y tiene un bajo índice de falsos positivos lo que da confianza en el sistema.

De este modo los usuarios no deben preocuparse que su servidor de mail devore sus mensajes importantes pensando que contienen algo desagradable o que son de una fuente spam.

El soporte disponible para GLMail es ciertamente muy bueno y diferentes contratos pueden ser comprados dependiendo de las necesidades de nuestro propio negocio.

Si usted está interesado en mirar a GLMail, puede bajarse un trial de 28 dias que está disponible en su web page. Con el filtrado de spam y el virus que tiene (lo cual generalmente no está disponible al mismo nivel con los productos de Open Source), GLMail está cerca de ser un "*killer application*".

¿Como funciona el webmail?

El presente artículo está extractado de un "working report" realizado en la Universidad de Pennsilvania, USA por el Webmail Evaluation Task Force en el año 2000, para evaluar la mejor solución para instalar en la Universidad. Quienes quieran ver detalles de tal evaluacion y del porque fue elegido IMP (www.horde.org) lo podrán hacer en http://www.upenn.edu/computing/group/webmail/2000/Penn_webmail2000_report.html#How

Webmail emplea una capa adicional entre el usuario y lo que llamaremos el "e-mail store" (tienda de e-mail, un e-mail server), esto es, un modulo web que toma su input (datos de entrada) del usuario a través de browser del cliente, parsea (en inglés "parsers") o interpreta ese input, y se comunica con el "e-mail store" y/o procesos IMAP/POP/SMTP. Los productos ofrecidos generalmente (aunque no siempre) operan con un mail-server externo, esto es, un mail server no integrado dentro de la aplicación webmail misma.

Los webmail servers normalmente pueden ser instalado en los mismos hosts (maquinas) que los mail servers (servidores de correo), pueden ser instalados en servers separados de los mail servers, y pueden, en algunos casos proveer una puerta de acceso a múltiples mail servers. Por ejemplo en el caso de Penn University se pudo implementar una solución donde cada mail server utilizado utilizaba su propia implementación de webmail, bien en su e-mail host o en un web host diferente. Alternativamente, Penn podía implementar una solución donde un webmail server central podría servir a varios sistemas de e-mail. No existe ningún requerimiento para que la maquina web corriera en el mismo host que la maquina de e-mail.

Los productos estudiados fueron:

IMP: http://www.horde.org (el elegido)



TWIG http://www.informationgateway.org/

WorldPilothttp://www.worldpilot.nl/WorldpilotSite/Worldpilot/Informatie

Otras posibilidades:

http://www.openwebmail.org/

http://www.squirrelmail.org/

Tabla comparativa

	Qmail	Sendmail	Postfix	Exim	SuSE	GLMail
POP	NO	NO	NO	NO	SI	SI
IMAP	NO	NO	NO	NO	SI	SI
Notas	NO	NO	NO	NO	NO	NO
Precio	GRATIS	GRATIS	GRATIS	GRATIS	$ 999	$ 525
SSL	NO	SI	NO	SI	SI	SI
LDAP	NO	NO	NO	SI	SI	SI
Listas	NO	NO	NO	NO	NO	NO
Web	NO	NO	NO	NO	SI	SI
Filtro	NO	SI	SI	SI	SI	SI
RBL	NO	SI	SI	SI	NO	SI
Virus	NO	NO	NO	NO	NO	SI
Puntaje General	6	7	5	9	8	9

Usando Perl

¿Qué es Perl? Perl es un lenguaje de scripting diseñado para procesar texto contenido en grandes cantidades de archivos. Debido a esta característica ha tenido una gran aceptación y hoy en día es ampliamente utilizado para la creación de scripts CGI.



Contexto

Comencemos poniéndonos en contexto. Si combinamos 2 premisas: el mundo ***GNU/Linux*** (en cualquiera de sus *flavors*, extensivo a cualquier *flavor* de **Unix**) y los lenguajes de programación, seguramente lo primero que se nos viene a la mente es el Lenguaje ***C***.

C es el lenguaje por excelencia para escribir (codificar) programas y/o aplicaciones de bajo nivel, de hecho, el *kernel* y la mayoría de las utilidades de ***GNU/Linux*** están escritas (codificadas) con él. Pero, dependiendo de las tareas que deseemos hacer, ***C*** no es la mejor opción, un ejemplo: la manipulación de contenidos de múltiples archivos. ¿Por qué este ejemplo? Porque para este tipo de tareas existen los lenguajes de *scripting*. Las capacidades de *scripting* de alto nivel que tienen las *shells* (intérpretes de comandos) de ***GNU/Linux***, combinadas con el uso de utilidades de bajo nivel las hacen excelentes para este tipo de tareas.

Hasta ahora mostramos dos extremos posibles. Por un lado, un lenguaje de programación, que una vez compilado tiene una altísima performance de ejecución, pero es difícil programar en él y además tiene performance dudosa manipulando contenidos de múltiples archivos. Por el otro están los lenguajes de *scripting,* que responden bien manipulando archivos, pero tienen una gran desventaja: no se compilan, se interpretan línea a línea, por lo consiguiente tienen una menor performance (son menos eficientes) en su ejecución.

El diseño de *Perl* no se basó en la elegancia, sino en la funcionalidad

La solución a este problema es encontrar algún punto intermedio entre estos dos extremos. Aquí es donde nacen un sinnúmero de lenguajes. Uno de estos lenguajes intermedios, por cierto bastante conocido, es ***awk***. ***Awk*** nació para reemplazar y mejorar las limitaciones de usar ***grep*** y ***sed*** dentro de *shellscripts*. Su sintaxis es similar a la de ***C*** y tiene mejores características que ***sed*** para manipular patrones de texto (*patterns*) dentro de un archivo.

¿Cómo nace Perl?

Siguiendo esta tendencia de lenguajes intermedios, de la mano de Larry Wall nace ***Perl*** (***P****rogrammable* ***E****xtraction and* ***R****eport* ***L****anguage*). Hace más de 10 años, Wall desarrollaba aplicaciones y utilitarios para la NSA (*National Security Agency*-Agencia de Seguridad Nacional de los Estados Unidos) y necesitaba agregarle funcionalidades de red al ***awk***. En vez de reescribir o modificar el ***awk***, tomó ideas y sumó entre sí características de lenguajes y utilitarios ya existentes, las más notorias son: capacidades de *networking*, control de flujo y sintaxis similar a ***C***, nomenclatura de variables estilo *shellscript* (uso de **$**, **%** y **@** precediendo nombres de variables), procesamiento de texto del ***awk***, rutinas de manipulación de cadenas de caracteres de ***Basic*** (por ejemplo funciones como *substr()* y *length()* para manejar cadenas de caracteres) y rutinas de generación de reportes de ***RPG-2*** (*Report Program Generator*). El diseño de ***Perl*** no se basó en la elegancia, sino en la funcionalidad, según propias palabras de Larry Wall: "*Una de mis prioridades era considerar que los lenguajes de programación se escriben para que las personas puedan usarlos convenientemente*". Como resultado, ***Perl*** no es tan rígido como otros lenguajes de progrmación (por ejemplo ***C*** o ***Pascal***). Es tan flexible que frecuentemente se cae en un dilema problemático: para realizar una tarea, *siempre hay más de una manera de hacerla*.

Características

Una característica curiosa, y a la vez importante, es que ***Perl*** no es un intérprete y tampoco es un compilador. Es un raro híbrido llamado "interpilador". Cuando se invoca a ***Perl*** para ejecutar un programa, el proceso de "interpilación" consta de tres partes:

1- Apertura de las fuentes, lectura completa del archivo y verificación de sintaxis (*parsing*).
2- Compilación del programa.
3- Interpretación del programa compilado.

Perl es un raro híbrido llamado "interpilador"

Gracias a los pasos 2 y 3, los programas en ***Perl*** tienen una performance de ejecución muy superior a los *shell-scripts*, debido a que estos últimos deben verificar e interpretar la sintaxis línea-a-línea.

Desgraciadamente esta ventaja tiene su contrapartida: debido a la existencia del paso 1, los programas en ***Perl***, tienen una performance de *start-up* (inicio de ejecución) muy inferior a los *shell-scripts* y a los programas compilados.

Crecimiento

En 1988 Wall liberó ***Perl*** como *freeware* (software libre), fue adoptado rápidamente y migrado a una gran variedad de plataformas, como por ejemplo, **Windows**, todos los *flavors* de ***Unix*** y ***Linux***, ***MacOS*** e inclusive *mainframes* de ***IBM***. Hoy en día, ***Perl*** es un componente vital del mundo ***GNU/Linux***, ya que muchas distribuciones lo incorporan para agilizar funciones de administración del sistema. ¿La Razón? Están implementadas casi la totalidad de las librerías de llamadas al Sistema Operativo (*System-Call Libraries*). A través del tiempo, fue ganando una muy buena reputación para realizar tareas de *scripting* en servidores web y tiene una muy buena integración con servidores como ***Apache***. Su funcionalidad más usada en esta rama son las ***CGIs*** (*Common Gateway Interface*). Actualmente ***Perl*** va por la versión 5.8.4 y esta en desarrollo la versión 5.9.1.

Futuro

Perl 5 fue lanzado en 1995 y, con unos cuantos años de historia en su haber, ya ha alcanzado un estado donde se vuelve difícil agregarle funcionalidades. Por esta razón, a mediados del año 2000, Larry Wall anunció que se iniciaba un borrador de proyecto para desarrollar ***Perl*** 6. En el desarrollo de ***Perl*** 6 habrá una gran participación de la comunidad de actuales desarrolladores, pero la metodología de aprobación e implementación de nuevas características será un poco dictatorial. De esta manera, analizando los requerimientos de los programadores de la comunidad, Wall y su selecto grupo de programadores dentro del proyecto tendrán la última palabra sobre lo que será o no será implementado.

En Perl siempre hay más de una manera de hacer las cosas

La idea principal sobre el nuevo ***Perl*** 6 se centra en cambiar (ligeramente) el concepto de "interpilador", haciendo que trabaje de manera similar a como lo hace ***Pyton***. De esta manera, el proceso de lectura del archivo fuente, la verificación de su sintaxis y su compilación a un *byte-code* intermedio se realizarán solamente con la primera ejecución de un programa perl. El *byte-code* intermedio no se descartará hasta que haya una nueva versión del código fuente que lo generó. Además, la porción "compilador" de ***Perl*** será capaz de emitir código ***C***, ***Java*** y ***C#***.

Un objetivo a lograr con ***Perl*** 6 es reestructurarlo de tal manera para que sea más sencillo extenderlo dinámicamente, logrando que, desde el punto de vista del desarrollo modular del lenguaje, sea más flexible. Se reacondicionarán los tipos de datos internos (*built-in data types*) y se le agregará la capacidad de OOP (*Object Oriented Programming*-Programación Orientada a Objetos). La suma de estas nuevas características hará que ***Perl*** sea más propicio para proyectos de desarrollo de software de gran escala.

Bibliografía

"Programming Perl", 3ra Edición
de Larry Wall et al
Editorial O'Reilly

"Perl Tutorial"
de Charlie Stros (publicado dentro de "The Complete Linux Handbook")
Editorial Future Publishing

¿Cómo es un programa en Perl?

Veremos como ejemplo un pequeño programa que permite al usuario ingresar texto y luego verlo reflejado en pantalla una vez finalizada la ejecución del programa. Se puede ingresar cualquier texto y para finalizar la ejecución debe ingresarse la palabra "terminar".

```
1.#!/usr/bin/perl
2.
3.$in = "";
4.@lines = ();
5.
6.print "Ingrese algun texto > ";
7.while($in = <STDIN>) {
8. print "\n";
9. last if ($in =~ /^terminar$/i);
10. push(@lines, $in);
11. print "Ingrese algun texto > ";
12.}
13.chomp @lines;
14.print "\n";
15.print "Usted ingreso: ";
16.
17.for ($i = 0; $i <= scalar(@lines); $i++) {
18. print sprintf("%03d", $i), $lines[$i] "\n";
19.}
20.
21.exit;
```

Paso a Paso

La primera línea es opcional pero se usa para evitar problemas. No es parte del programa, ya que su única funcionalidad es informarle a la *shell* que lo que sigue es un programa en ***Perl*** y que debe invocarse al intérprete de ***Perl*** que esta ubicado en ese *pathname*. (Si el intérprete de ***Perl*** esta ubicado en otro lugar, por ejemplo en **`/bin/perl`** ó **`/usr/local/bin/perl`**, debe cambiar esta primera línea para que concuerde).
En las líneas 3 y 4 se inicializan dos variables.
A diferencia de ***C***, que tiene tipos de variable "entero", "punto flotante" y "caracter", ***Perl*** solamente tiene tres distinciones para las variables: singular, plural y diccionario.
Las variables singulares, llamadas "*scalar*", se reconocen con un signo $ que las precede. Y pueden almacenar cualquier valor (numeros, letras, etc.) y todo lo almacenado dentro de ellas será considerado una unidad de datos.
Las variables plurales y diccionarios son dos variaciones de *arrays*. Los plurales, llamados "*array*", se distinguen con el signo @ y corresponden a *arrays* que tienen valores que están indexados numéricamente. Los diccionarios, llamados "*hash*", se distinguen con el signo % corresponden a *arrays* que están indexados por valores, es decir, en vez de tener un subíndice numérico, se pueden utilizar subíndices de caracteres o palabras, dependiendo de la necesidad o criterio del programador.

En la línea 6 solicitamos al usuario ingresar algún texto.
En la línea 7 iniciamos el bucle de repetición, vemos que la estructura del bucle es muy similar a ***C***. **`STDIN`** representa el ***ST****andar****D IN****put*, la consola. El bucle se repetirá hasta que no exista más la entrada por defecto, como esto no sucederá, en la línea 9 cortamos el bucle con la sentencia **`last`** que evaluará la condición de existencia de la palabra "terminar" en la entrada recibida del usuario.
En la línea 10 vemos cómo se agrega un *scalar* (**`$in`**) a un *array* (**`@lines`**).
En la línea 13 se ve la utilización del comando **`chomp`**, que quitará de cada ítem del *array* los "Enter" (*carriage return*).
De la línea 17 a la 19 vemos la sintaxis de otro bucle de repetición, la sintaxis es idéntica a la sentencia **`for`** de ***C***. dentro del bucle vemos el uso de la función **`sprintf()`** que funciona tal cual a la de ***C***.
Finalmente en la línea 21, finalizamos expresamente la ejecución con la sentencia **exit**.

Perl en la Web - Direccionario

Toda la información y recursos disponibles sobre ***Perl***, estan sostenidas por 2 grandes pilares: La Editorial O´Reilly e Internet.
Por más raro que suene, la editorial O'Reilly le paga un sueldo a Larry Wall para que se encargue, a tiempo completo, de mantener el desarrollo de ***Perl***. La editorial no sólo es la que se encarga de publicar los mejores libros de referencia sobre el asunto: "*Programming Perl (3rd Edition)*", "*Learning Perl*", "*Perl in a Nutshell*" y "*The Perl 5 Pocket Reference*"; sino que además, mantiene el portal oficial de ***Perl*** en Internet: *http://www.perl.com*.



El portal sirve como punto de partida para todo programador de ***Perl***, sumando entre sus contenidos toda la documentación on-line y un link a ***CPAN*** (*Combined Perl Archive Network*), la fuente de casi cualquier *download* de módulos y código fuente de ejemplo para ***Perl***.

Otro sitio de consulta, en Internet, para ***Perl*** es: *Perl Mongers* (*http://www.perl.org* y *http://perlmonks.org*), una organización de programadores de ***Perl*** que sirve como punto de convergencia de grupos locales de usuarios y programadores de ***Perl***. Si está buscando una manera rápida de aprender ***Perl*** ó alguna solución ya hecha para una tarea que necesita resolver, este sitio es un buen punto de partida.

USERS

#01

.code

COMUNIDAD DE DESARROLLADORES

El Futuro de .net

Un recorrido por Whidbey y Yukon: características principales, cambios, mejoras y todo lo que hay que saber sobre Visual Studio 2005 y SQL Server 2005.

CUSTOM CONTROLS para ASP.NET | Cómo conectar a web services de terceras partes | PHP: funciones de indexación y búsqueda para un portal

DATABASE Optimización de bases y consultas en SQL 2000 | Persistencia y OR Mapping con C# y ASP.NET | Datasets tipados desde VB.NET

DIRECTX en .NET con C# | Introducción a Managed DirectX | Inicialización de Direct3D y dibujo de objetos | Composición de escenas y movimientos

ADEMAS, JAVA Y JDBC | Pathfinding: búsqueda de caminos con el algoritmo A* | Noticias | Reviews: Rent A Coder – NET Reflector – Dynamic PDF

WHITE PAPER: ORIENTACION A OBJETOS

Small Business Server 2003

La mayoría de las pequeñas empresas comienzan con un par de computadoras personales conectadas en una red básica "peer-to -peer" con una conexión compartida de la impresora y de Internet. A medida que crecen, sin embargo, tales empresas desearán a menudo aprovechar servicios, tales como calendarios compartidos, e-mail administrado y acceso remoto. Para eso necesitan una red "cliente-servidor" más sofisticada.

Desafortunadamente, crear una red de una pequeña empresa, basada en un modelo "cliente-servidor" nunca fue fácil. De tal modo, para las pequeñas empresas, especialmente las que tienen veinte empleados o menos, la última versión de Microsoft Small Business Server (SBS 2003) es una excelente opción, siendo considerablemente más fácil de configurar y mantener que las versiones anteriores. SBS 2003 es promocionado por Microsoft como la "solución completa para servidores de negocios en una caja".

Se construye alrededor de tres aplicaciones básicas: e- mail y calendario, backup de archivos y servicios de archivos e impresoras. Entre sus atracciones, permite a los usuarios acceder a sus archivos personales y a e-mails desde y hacia cualquier PC conectada a la web, y sincronizar sus dispositivos móviles tales como notebooks y teléfonos inteligentes "Microsoft-powered".

Otra característica es que incluye una versión de Microsoft Share Point integrado para trabajo colaborativo y un servidor para enviar y recibir faxes.El software viene en dos versiones.

- La edición Estándar, que incluye un Windows Server 2003, Exchange Server 2003 y Outlook 2003, junto con las cinco licencias cliente de acceso que usted necesita para cada computadora (o para cada usuario), cuesta U$S 1699.- + IVA
- La edición Premium, a U$S 4699.- + IVA que añade las bases de datos de Microsoft SQL Server 2000, ISA Server 2000 para seguridad y funciones integradas de servidor proxy, y FrontPage 2003 para diseño web. Versiones anteriores de SBS eran notoriamente difíciles de configurar (como en la versión 3.5).

Ambas versiones se venden en este momento con un antivirus de McAfee gratis para servidor y 5 usuarios.

Esta vez Microsoft guía a los usuarios a través de la configuración con un cuadro de "comienzo rápido".

Esta es una maniobra de bienvenida, pero los dueños de pequeñas empresas aún pueden encontrar un poco difícil el proceso. Para ser justos, SBS 2003 no está propuesto para los novatos en redes.

Muchas empresas pequeñas compran a un revendedor, una PC de bajo costo con SBS preinstalado. Pero si usted decide empezar desde cero, una configuración típica de SBS Standard con hardware costaría alrededor de $ 4300.- y le llevaría casi cuatro horas configurarlo.

Para correr SBS 2003, Microsoft recomienda como mínimo un procesador de 550MHz, 384Mb de RAM y 4Gb de espacio libre en disco, pero es recomendable elegir una máquina más poderosa.

La primera vez que usted reinicia el servidor después de instalar SBS 2003, usted o el ayudante técnico que haya contratado, estarán buscando un cuadro de diálogo que le presente la pregunta: ¿El servidor esta conectado directamente a Internet o a través de un router?

Configurar una PC cliente es fácil, sin embargo, sólo corre automáticamente en Windows 2000 y XP, las PCs que tienen instaladas versiones anteriores de Windows, tendrán que ser configuradas manualmente.

Luego de darle a SBS los nombres de las computadoras a configurar, usted abre un navegador en cada máquina cliente, la direcciona hacia una página en el servidor SBS 2003 interno y sigue las instrucciones para conectar al cliente con el dominio.

Cuando se "loguea" en el servidor web desde el cliente, las aplicaciones y "service packs" son *empujados* hacia ese sistema cliente. Una vez que el proceso de configuración está completo, una lista de tareas lo guía a través de los pasos restantes, que incluyen obtener un certificado de acceso seguro, configuración del firewall, y otras opciones de seguridad.

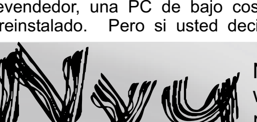
Cuando el servidor está funcionando, se puede acceder a todas sus características desde las pantallas de configuración y mantenimiento, incluyendo la pantalla "home server" que permite a los usuarios configurar nuevas PCs cliente y administrar las conexiones a e-mail, Internet y opciones de back up.

Vemos un SBS 2003 estable y digno de confianza.

Entre sus más destacadas prestaciones están la del calendario compartido, que permite compartir tareas entre los miembros del equipo, y el acceso y administración remotos.

Usted (o el administrador de la red) puede conectar cualquier PC con Windows XP a la red utilizando la característica de escritorio remoto de cualquier navegador.

Igualmente impresionante es lo siguiente, si usted está en un viaje de negocios puede acceder a su e-mail y sincronizar dispositivos móviles. En conjunto, Microsoft ha hecho grandes avances con SBS 2003 al cubrir la necesidades de los usuarios de pequeñas empresas que carecen de recursos tecnológicos dentro de la compañía, aunque muchos de ellos todavía necesitarán hacer las cosas a mano.

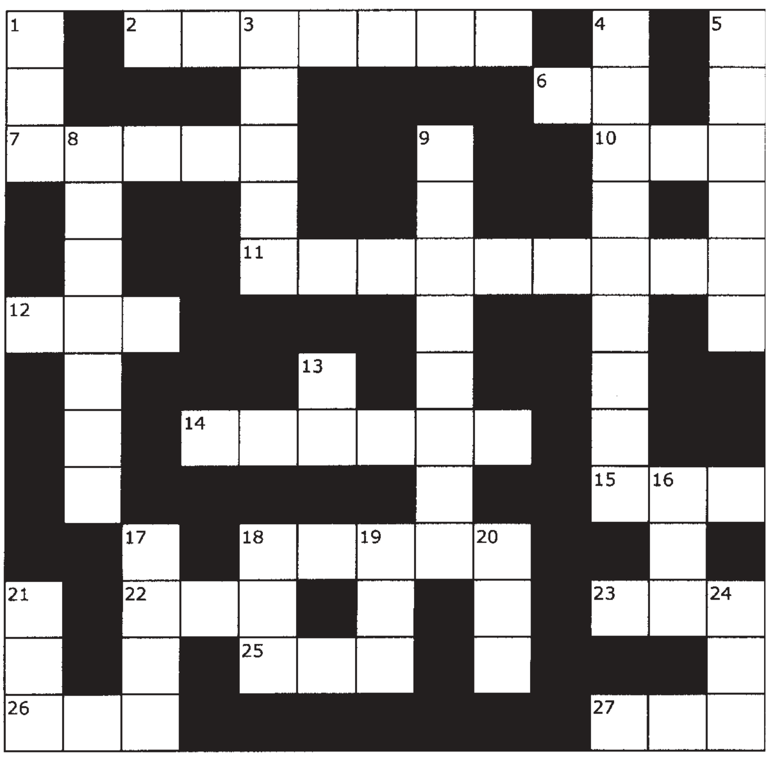


Nvu es el nombre de un nuevo programa de diseño web para Linux que combina funciones de los populares Frontpage y Dreamweaver.

El programa está siendo desarrollado por el equipo de Michael Robertson (responsable de la distribución Lindows.com), es de licencia libre y sus funciones son similares a las que podemos encontrar en las herramientas de diseño Frontpage y Dreamweaver.

"Ahora cualquiera puede crear una página web, aunque no sea experto o conozca el lenguaje HTML", podemos leer en la página web Nvu.com del programa.

Actualmente está en estado de desarrollo, aunque entre las funcionalidades que han sido anunciadas destaca la integración de FTP, modo WYSIWYG y código, soporte de frames, formularios, tablas y plantillas. El programa estará listo para 2004.



Horizontales

2. Aparato que transforma cualquier documento gráfico que esté en papel o película, en un archivo de datos digitales que puede ser revisado e incluso modificado, mediante un computador (en inglés)
6. Extensión de archivos de Adobe Illustrator
7. Lenguaje de programación inicialmente diseñado para principiantes (Beginners All-purpose Symbolic Instruction Code)
10. Extensión de archivos de CorelDraw
11. Dispositivo periférico que reproduce textos e imágenes en papel
12. Es la extensión que corresponde a un tipo de fichero ejecutable bajo MsDos
14. Uno de los más populares programas cliente de correo electrónico. Fue desarrollado por la empresa norteamericana Qualcomm. Es una palabra griega que significa "buena piel"
15. Extensión de archivos de texto plano
18. Dispositivo electrónico de pequeño tamaño, dotado con teclas (y a veces un bola, llamada trackball), operable con la mano y mediante el cual se pueden dar instrucciones al ordenador para que lleve a cabo una determinada acción
22. Siglas en inglés para pantalla de cristal líquido, usada generalmente en las notebooks y otras computadoras pequeñas
23. Extensión de archivos de ayuda en línea de programas (Windows Help)
25. Extensión de archivos de PaintBrush (Windows)
26. Siglas en inglés para Red Privada Virtual
27. Abreviatura en inglés para dígito binario. Es la unidad más pequeña de almacenamiento en un sistema binario dentro de una computadora

Verticales

1. Concentrador. Dispositivo que se utiliza típicamente en topología en estrella como punto central de una red, donde por ende confluyen todos los enlaces de los diferentes dispositivos de la red
3. Estándar aceptado casi mundialmente que recoge 128 caracteres, letras, números y símbolos utilizados en procesadores de textos y algunos programas de comunicaciones. Su principal ventaja es su amplia difusión y aceptación. De hecho, la mayoría de los procesadores de textos presentes en el mercado pueden importar y exportar ficheros a este formato , lo que facilita el intercambio de información entre personas o empresas que no trabajan con la misma aplicación
4. Empresa credora del S.O. Windows
5. Método para transmitir datos secuencialmente, es decir, bit por bit
8. Software de intercambio de documentos de Adobe Systems, Inc., Mountain View, CA, que ejecuta sobre DOS, Windows, UNIX y computadores Macintosh
9. Protocolo de autenticación que permite intercambiar login y password de manera encriptada, entre el cliente y el servidor
13. Nomenclatura estándar de Disco Rígido
16. Meta-lenguaje que permite definir lenguajes de marcado adecuados a usos determinados. En la práctica corresponde a un estándar que permite a diferentes aplicaciones interactuar con facilidad a través de La Red
17. Ordenador personal de características, funcionalidad y rendimiento similares a los PCs de fabricantes de prestigio mundial, pero que no cuenta con una marca reconocida internacionalmente
18. Extensión de archivos de bases de datos (Microsoft Access)
19. Protocolo del nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya esta blecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Se utiliza cuando se necesita transmitir voz o video y resulta más importante transmitir con velocidad que garantizar el hecho de que lleguen absolutament e todos los bytes
20. Es la extensión que corresponde a un tipo de fichero ejecutable bajo MsDos, OS/2 y/o Windows
21. Extensión de archivos de sonido (Media Player)
24. Extensión de archivos de Carpetas personales de Microsoft Outlook